-ACTAS-
ADAPTIVE COMPOSITION AND TRADING WITH AGENTS FOR SERVICES

SUBMISSION FOR
DOCTOR OF PHILOSOPHY

by

Reinhold Kloos

Diplom-Informatiker

E-Mail: r.kloos@soi.city.ac.uk

April 2013

Department of Computing
School of Informatics
City University London

Institute for Computer Science and
Business Information Systems
University of Duisburg-Essen

Supervisor: Prof. Dr. Rainer Unland, University of Duisburg-Essen
Assistant Dean: Dr. Peter W.H. Smith, City University London

CONTENT

FIGURES

TABLES

ACKNOWLEDGEMENTS

I am most indepted to my supervisors Prof. Dr. Rainer Unland and Prof. Dr. Michael Schroeder for their support. Sincere thanks I give to Prof. Bernhard Cohen, who was my second supervisor and inspirator for a long period of my research.

I enjoyed working with my colleages and fellow students Tshiamo, Penny, Panos, Panayiota, Aloysius, and Marcus. Their motivation kept me going. My gratitude goes especially towards Dr. Swen Wagner, Dr. Jacek Gomoluch, Dr. Dominik Stein, and Dr. Stefan Hanenberg, who gave me valued advice and help through proof reading as well as other phases of my disertation. The support of Dr. Andrew Tuson, Marc Firman, Eric Smith, Dr. Eddy Parkinson, Veronika Muntoni, Christina Braun, Bernd Göhing, and Gottfried Merkel, I will never forget.

Publications of my research are [KlUnBr2009], [KlUnBr2010], and [KlUnBr2012]. Early publications of my research have been [KlReSc2002], [KlReSc2001], [KlHoSc2000], and [KlScRe2000]. I started my research in software agents and services as a research fellow at the institute of Secure Telecommunication of the Fraunhofer society (FhG-SIT, former GMD-SIT) in Darmstadt, Germany. The declarative environment of ACTAS is based on my research with Prof. Dr. Christoph Beierle for my academic title as a Computer Scientist with diploma at the University of Hagen (cf. [BeKlMe2000]). The work was sponsored through two scholarships: a one year program by the German Academic Exchange Service (DAAD) and a three year funding by the City University London.

My thesis is dedicated to:

My Father	**My Mother**
Otto Franz Alban Kloos	**Annemarie Kloos, née Karbaum**
23.04.1934 – 25.01.2000	**03.06.1929 – 04.07.2012**

My Godfather

Georg Karbaum, who died 2006

Declaration

ABSTRACT

Mainly in business domains, the vision of gaining flexible, adaptive service environments is based on the standardization and practical proliferation of (Semantic) Web Services, ontologies, and agents. The standards of Web Services and their Service-oriented Architectures (SOA) became the standard paradigm for software component integration. Dynamic changes and the permanently increasing amount of available e-services of different domains are a challenge of Service Discovery and Composition. Mediation between different approaches and expert knowledge is often necessary for the composition of services of different domains. Semantic enhancements, Autonomic Service Discovery, and the research for more holistic concepts for the classification of e-services are current attempts of overcoming this challenge, in order to reach the ultimate goal of Autonomic SOC.

The thesis introduces concepts and models of a Service Discovery framework called ACTAS (Adaptive Composition and Trading with Agents for Services). Aware that an absolute definition of services does not exist, a classification of e-services was developed on the base of four different aspects of services. Through these aspects, so-called Semantic Characteristics are commonly agreed. As “building blocks” Semantic Characteristics ease the Service Description, Service Discovery, and Service Composition. Services and requests holding the same Semantic Characteristics are principally compatible. In its semantic context, a Semantic Characteristic can declare some properties through also commonly agreed Property Classes. These specific classes of ACTAS enable the integration of algorithms into the declarative environment for the checking of constraints and mediation of data. The constraints can check the plausibility of property values before and after Service Composition. Additionally, Exchange Constraints can interlink properties of different Semantic Characteristics. ACTAS propagates the publications of Semantic Characteristics and Property Classes. A Multi-Agent System (MAS) builds the middleware of ACTAS supporting the domain specific policies of Service Provision, Service Trading, and the incorporation with the service requesting application environment.

ABBREVIATIONS

B2B	Business-to-Business interface
B2C	Business-to-Customer interface
BPEL, BPEL4WS	Business Process Execution Language for Web Services
BPMs	Business Process Management Systems
DAML	DARPA Agent Markup Language
DFC	Distributed Feature Composition
DIS	Distributed Information System
DL	Description Logics
DOC	Distributed Object Computing
DSL	Domain-Specific Language
EAI	Enterprise Application Integration
FDL	Feature Description Language
FOL	First Order Logic
OWL-S	Ontology Web Language for Services
SAWSDL, WSDL-S	Semantic Annotations for WSDL (Web Service Description Language)
SLA	Service License Agreement
SOA	Service-Oriented Architecture
SOC	Service-Oriented Computing
(S)WS	(Semantic) Web Service
SWSF	Semantic Web Services Framework
WSMO, WSML	Web Service Modelling Ontology / Language
WfMs	Workflow Management Systems

ACTAS ABBREVIATIONS

ASO	Actor Service Offer	PC	Property Class
AST	Actor Service Template	RCh	Request Characteristic
CCh	Compatibility Characteristic	ReA	Request Agent
Char	Semantic Characteristic	RM	Request Mode
CoA	Composition Agent	RP	Request Port
CompSt	Composition (Service) Structure	SM	Service Mode
CompSt-plus	Composition (Service) Structure enhanced	SO	Service Offer
Ex-Co	Exchange Constraint	SOER	Service Offer Export Record
ExName	Exchange Name of Exchange Constraint	SP	Service Port
Env	Environment (Value Constraints, Exchange Constraints, and Option-Slots)	ST	Service Template
		SRe	Service Request
FA	Facility Agent	TrA	Trader Agent
GCh	General Characteristic	TRe	Trading Request
Me-Co	Merge Constraint (Mediation and Matching)	Va-Co	Value Constraint
PA	Personal Agent		

INTRODUCTION

1 Motivation

We live in a world of services. Services performed by agents on behalf of others are part of our daily life. A service is sensed as an asset with an inherent value and its consumption normally involves the transfer of value and the generation of cost with the consequent need of its settlement. In business for instance, supply chains demanded early on the discovery, composition and planning of services through electronic services (e-services). In fact e-business became a main area for the application of the service paradigm. Nevertheless, the development towards a web of services will have its implication on many disciplines including associated ones like social services and psychology for the support of the security of data and an improved observation of the privacy of people.

Generally, a Service Requester can request services offered by Service Providers. The Service Requester does not need a complete knowledge about the services; he[1] has to rely and trust on the established interfaces and policies of the Service Providers. Business Services distinguish relationships between a service and Service Client (B2C) as well as between two services (B2B). The latter relationship follows matching rules determined by internal policies of businesses. Other domains like Computer Supported Cooperative Work (CSCW) had their own ideas about services; they thought of several Service Clients demanding Technical Services. The transparent establishment of a communication service between two Service Clients with the currently available communication facilities is an example of such a Technical Service. The term “Web 2.0”, introduced by [Kno2003], is used in the context of services for social networking and Cloud Computing. The intent of the latter is quite similar to Technical Services of CSCW approaches. However, it is combined with ideas of Business Services, because Cloud Computing propagates a specifically scaled billing of computing resources offered in several ways.

The number of services, which will be offered on the Internet, is expected to rise dramatically in the next few years. Technologies like Semantic Web [W3C2009a; W3C2009b] or Web Services [BooHaa et al.2004; IBM2009] transform the Internet from a network of information to a network of knowledge and services, generating dreams of Web3.0; also called **Autonomic Service-Oriented Computing** [RamHol et al.2009; ToDePe2009], i.e. the reduction of the requirement of manual, human intervention in service-oriented software scenarios including learning of the users’ preferences. Research Projects, like e.g. Service Web3.0 [Eur2009; Ser2009], drew already the picture of a new generation of the World Wide Web after Web2.0. Interestingly, Tim Berners-Lee, inventor of the World Wide Web, questioned whether one could use the term Web2.0 in any meaningful way, since many of the technological components of Web2.0 had

[1] The use of a specific pronoun shall not imply any discrimination of gender.

existed since the early days of the Web[Ars2006].[IBM2006]. He strongly propagated the ideas of Semantic Web and the web of services [Ber2003; TalWor2008], which led to the research of Semantic Web Services (SWS) [BrzRek et al.2010; KlUnBr2010].

1.1 The challenges of Service-Oriented Computing (SOC)

Two main challenges appear with Service-Oriented Computing (SOC): (1) the integration challenge and (2) the semantic challenge. The integration challenge describes the problem to integrate and to compose services running on different systems. For the tackling of the integration challenge, Business Services developed the multi-layered middleware of Distributed Information Systems (DIS). The DIS approach was soon extended to an Enterprise Application Integration (EAI), which enriched the composition mechanism with business logic like the workflow management or petri nets. However, the developed environments for Service Composition suffered under the integration problem due to the lack of standards.

The standards of Web Services changed the situation. They described Service-oriented Architectures (SOA), offering Service Trading, Service Composition, and Service Coordination. The goal of the latter was a coordinated starting of services in the phase of service deployment (Deployment Phase). Phases of a life cycle of a Composite Service consisting of several Component Services can be introduced in general: Service Design, Service Discovery, Service Composition, Service Grounding, Service Deployment, and Service Consumption/Execution. Component Services can be Abstract Services, which possibly enforce an anew Service Discovery for Concrete Services, which can be deployed. Services can have an inherent complexity, i.e. their orchestration (the description of the Component Services called for the realisation of one Composite Service) and their choreography (the possible protocols of message exchanges between the Component Services) can be hidden or transparent to Service Clients due to company policies for instance. In Software Engineering, Service-oriented Architectures became a standard paradigm for software component integration besides distributed objects and agents. Although these terms are used in so many domains in various contexts with different purposes, there is no commonly agreed and standardised distinction of the entities.

The second commonly mentioned challenge of Service-Oriented Computing is the semantic challenge. In the standards of Web Services, the description of methods or messages is done in the Web Service Description Language (WSDL). Depending on the model of the service invocation, WSDL supports the synchronic call of methods in the style of Remote Procedure Calls (RPC) or the asynchronic, message oriented invocation of services. The messages and methods describe parameters. The semantic challenge is to overcome the heterogeneity and ambiguity of the data presented by the parameters, in order to avoid semantic mismatches. The resulting research area of Semantic Web Services (SWS) combines techniques of Semantic Web with Web Services. The best known approaches of Semantic Web are Ontology Web Language for Services (OWL-S) [W3C2009a] and Web Service Modelling Ontology (WSMO) [BruBus et al.2005].

SWS approaches do not describe the functionality offered by a service through its messages, operations, or orchestration, but through its **service capability**, which contains semantically enhanced elements. The semantic classification of the elements used in a capability description is done through the application of ontologies. Both Semantic Web Service standards (OWL-S and WSMO) define or allow different kinds of capability descriptions. The mostly used capability description in the context of OWL-S is called IOPE standing for Input, Output, Pre-Conditions, and Effects.

The descriptions of (Semantic) Web Services are informal. That is well intended because in this form, the Service Descriptions can be designed and exchanged in the World Wide Web network easily. It needs approaches for the Service-Oriented Computing, in order to deal with the information of these Service Descriptions properly, i.e. according to given policies of the requesting application environment. New challenges for Service Discovery and Service Composition arise due to dynamic changes of Service Descriptions as well as the permanently increasing amount of available e-services of different domains in various semantic contexts. For instance, the distinct laws in countries can define different semantic contexts for the consumption of a service. Mediation between different approaches of SOC demands an expert knowledge about their various Service Descriptions. Even the number and versions of standards for Web Services increased in such a way, that standards were developed, in order to determine compatible standards and their versions for the service deployment.

Therefore, the earlier mentioned goal of Web3.0, the Autonomic Service-Oriented Computing, is still a long distance ahead. For achieving the goals of Autonomic SOC, several initiatives exist, in order to create comprehensive frameworks that integrate the vision of SOA and SWS: METEOR-S [VerGom et al.2005], IRS [CabDom et al.2006], and SESA [FeKeZa2008]. However, it might be that the vision can never be reached due to the inherent complexity of Composite Services, i.e. the various involved interests, policies, and laws can lead to challenges for the orchestration and choreography of Composite Services, which cannot be solved in general. Nevertheless, many of these challenges of dealing are often already solved by some domain specific approaches. Autonomic Service-Oriented Computing should take advantage of these existing approaches. Thus, it might be a good idea of Autonomic SOC to concentrate on the Service Discovery and Service Composition, in order to discover Service Providers offering Candidate Services, which promise firstly to match the Service Request and secondly to be composable with the other Candidate Services. The subsequent phases of the life cycle of the found services should take advantage of the mentioned domain specific approaches for doing further Service Composition, Service Grounding, and Service Deployment. Brokering concepts like [PaDaDi2010], following the just developed idea of Autonomic SOC, are discussed in the State of the Art chapter. Other approaches propose a context-based Composition Process (ConWeSc [SaNaMa2005]).

1.2 Introduction of ACTAS

The thesis introduces ACTAS – Adaptive Composition and Trading based on Agents. ACTAS is not another complete architecture supporting the whole Life Cylce of a service. It does not even propagate another complete functional description of a service. ACTAS suggest the introducion of a classification of services, in order to select services through their categories. It promises the early exclusion of non-matching services in the phases of Service Discovery and Service Composition through the classification of the services and the use of approved algorithms. The algorithms are used, in order to check constraints valid in the semantic context of a certain category of services. Services are selected and declared as principally compatible through the categories, they belong to. ACTAS might only discover and compose Candidate Services, but the adaptable discovery and composition should provide alternative Service Providers and information for the subsequent phases of the life cycle of services.

ACTAS is based on a Multi-Agents System (MAS), which uses the pro-activeness of the agents, in order to integrate the roles and policies involved in the SOC. Service Providers are integrated through Facility Agents (FA). The Service Requester, originated in the application environment, is represented by the Request Agent (ReA). The ReA create an agent for the Composition Process (CoA). In this way the Composition Process can be adapted to the policies of the application environment. A Trader Agent (TrA) can discover and compose services following its own policies. Due to its pro-active behaviour, the TrA could be also used for the integration of existing trading approaches. ACTAS introduced the role of a Service Client, in order to support Technical Services. A Service Client can be represented through a Personal Agent (PA). The MAS of ACTAS builds the middleware for the declarative environment of ACTAS. The declarative environment realizes Service Discovery and Service Composition through a backtracking based on approved algorithms. In the thesis, the declarative environment for the realisation of the Composition Process is described through the Composition Model (C-Model).

Similar to WSMO, which distinguishes between the concepts "service" and "goal", ACTAS presents separate models for the description of services (Service Model, S-Model) and requests (Request Model, R-Model). ACTAS is not restricted on Web Services, but e-services of any domain could be addressed through the Service Descriptions and Requests of ACTAS. The classification of services in ACTAS is defined through a special kind of ontological concepts, the so-called Semantic Characteristics, which integrate several aspects of services in their semantic description through relationships to ontological concepts of various ontologies. Aware that an absolute definition of services does not exist, a classification of e-services through four different aspects was developed: (1) the domain of a service, (2) its non-functional properties, (3) its inherent complexity, created through policies, orchestration, and choreography, as well as (4) its categorization in the phases of the service life cycle.

Commonly agreed Semantic Characteristics define a kind of quasi standard for the categorization of services. On this basis, the Semantic Characteristics can be used for the selection of services. A set of Semantic Characteristics addresses all services, which belong to the category described through the intersection of the categories of the Semantic Characteristics enumerated in the set. As "building blocks" Semantic Characteristics ease the Service Description and Service Composition. Services holding the same Semantic Characteristics (so-called Compatibility Characteristics) are principally compatible. ACTAS distinguishes between B2C (Business-to-Client) and B2B like Service Composition. The B2C like Service Composition, i.e. the (principal) compatibility between Service Request and Service Offer, is done through a specific kind of Compatibility Characteristics named Request Characteristics that can be related with ontologically defined user groups. ACTAS supports directed Service Composition, which define a server and a client side for the composition as it is usual in B2C relationships. Non-directed relationships, which can occur in the Service Composition of Technical Services, are also covered by ACTAS.

Semantic Characteristics define additionally a semantic context for properties (so-called Char Properties). Service Properties, i.e. properties of Service Descriptions in ACTAS, are always declared as Char Properties in the semantic context of a Semantic Characteristic. Similar to the association of a mediation algorithm with the mediator concept in WSMO, the Char Properties are associated with algorithms for the information handling, the matching, and the mediation. The research of ACTAS was concerned with the vision to take advantage of given algorithms. The Semantic Characteristics allow an application of these algorithms in a fitting semantic context. The algorithms are described through so-called Property Classes.

ACTAS defines three different kinds of constraints on the base of the algorithms associated with properties through Property Classes: Value Constraints, Merge Constraints, and Exchange Constraints. Algorithms for the handling of the information of a Char Property can be used for the definition of Value Constraints, in order to describe for instance domain specific restrictions valid in the Semantic Context of the Semantic Characteristic. Merge Constraints define the application of matching algorithms on the level of the Service Properties. So-called Exchange Constraints realize the mediation between Service Properties of different Semantic Characteristics.

ACTAS advocates the publications of Semantic Characteristics and Property Classes, in order to have commonly agreed concepts for the categorization of services as well as in order to have proven and effective algorithms for dealing with the properties. The management of these repositories as well as the association of algorithms with Char Properties could be the task of ACTAS Administrators. A Service Designer uses the Semantic Characteristics for the Service Description. A Service Request can be designed with the Semantic Characteristics and fitting constraints.

The association of Property Classes with the Char Properties ensures fitting formalisms for the various matching and mediation algorithms on the one hand. On the other hand, there is no need for the development and searching of fitting algorithms for given formalisms of the Service Description including non-functional attributes. Mediation algorithms can be applied more precisely, in order to translate from one kind of formalism to another one. In principle, ACTAS extends the idea of Semantic Web Services, which tackled the semantic challenge through the use of semantically enhanced capability specifications of services. For this purpose, the parameters of the functional description were ontologically classified. ACTAS classifies the services and uses the resulting categories of services for the Service Discovery and Service Composition. Additionally, approved algorithms associated with the Service Properties of the discovered services are used for constraints.

1.3 Structure of the thesis

The thesis starts with a description of the State-of-the-Art of services and discovery frameworks. The State-of-the-Art is parted in four parts. Chapter 2 has a closer look at a service and Service-Oriented Computing (SOC). It compares the classical idea of services with e-services and develops four aspects for the classification of services. The chapter 3 considers the approaches for solving the challenges of integration in the domains of e-business and software engineering. Web Services standards and Service Compositions are discussed. The Semantic Challenge for SOC is covered in chapter 4, which rely on ontologies. Capability Descriptions, i.e. semantically enhanced functional descriptions, are introduced, which are used in some of the covered approaches of Semantic Web Services. The introduced aspects of services structure the discussion of developments of Autonomic SOC in chapter 5. Holistic approaches with Semantic Web Services Execution Frameworks are sketched. The State-of-the-Art ends with the problem statement in chapter 6.

The description of ACTAS starts with the hypothesis (chapter 7) and the introduction of the System Environment (chapter 8). In the subsequent chapters 9 and 10, the Service Model (S-Model) covers the Semantic Characteristics and Property Classes as well as the Service Description of ACTAS (chapter 10). The perspective is changed from the Service Provider to the Service Requester with the Request Model (R-Model) introduced in chapter 11. The S-Model and the R-Model deliver the informal data structures of ACTAS.

The Composition Model (C-Model), covered in chapters 12 and 13, defines a declarative environment with data structures holding so-called Property Objects. Property Objects provide the handle to the algorithms associated with the properties through the Property Classes. The C-Model and the scope of the thesis are restricted to the Composition Process. The assumption is made that fitting algorithms can be integrated into the declarative environment as for instance Prolog modules. Other implementation instances of the algorithms could be accessible through (statefull) Web Services. It was decided that the implementation of the MAS, the provision of Property Classes, and the inclusion of their algorithms are not part of the current work of dissertation. The evaluation in chapters 14 - 18 shows with various scenarios the feasibility of the models of ACTAS. In the introduction of the models, examples are given with Technical Services, in order to include non-directed Service Composition. The thesis concludes with future research (chapter 22).

STATE OF THE ART

2 Service-Oriented Computing

In the last two decades, the (electronic) service (e-service) paradigm became dominant as a presentation of remote applications and as a modular entity of Software Engineering in distributed computing. The hype of **Service-Oriented Computing** (**SOC**) is based on **Web Service** (**WS**) technology and its idea of **Service-Oriented Architecture** (**SOA**) for a web-based, modular implementation of complex, distributed software.

In relation to Web 1.0, which is based on protocols and languages like HTTP, TCP/IP and HTML, the term Web 2.0 (especially in context of social networking and Cloud Computing), was introduced [Kno2003] with WS as a universal, standards-based integration platform. Besides the integration of applications, WS technology increasingly integrates a standardised publishing, transparent accessing, and specifically scaled billing of computing resources as they are propagated through grid [GuZaRo2009] and Cloud Computing [FuHao et al.2010; GraMax et al.2010]. Also real-time computing built with Web Service technology became a subject of recent research [HuaZha et al.2010; LiGaSh2010; TsaSha et al.2010]. One goal of the on-going research is **Autonomic Service-Oriented Computing** [RamHol et al.2009; ToDePe2009], i.e. the reduction of the requirement of manual, human intervention in service-oriented software scenarios including learning of the users' preferences. This is often phrased as Web 3.0.

E-services allow us the access of increasingly complex distributed software systems and resources of formerly often separated, heterogeneous domains through a standardised architecture. However, a commonly agreed idea of a service does not exist. Therefore, it is necessary to have a closer look at the aspects of services and terminology, in order to achieve the goals of autonomy.

2.1 Service

In our everyday life, we profit from services, which we perform ourselves, or which are performed on our behalf. Such Manual Services are specified in Definition 1. The service paradigm and it various aspects has been topic of research for decades [Kot1988, Kot2003; LaPaSt2003; Lov1983, Lov1988; O'EdHo2002]. The need to describe a service is analogous with the requirement for labelling goods or products. In comparison with a good, researchers tried to define unique characteristics of a service. Definition 1 mentions four common ones. As Table 1 shows, these characteristics did not hold in all use cases, since the service paradigm differed depending on the point of view of the researcher and the service domain.

Definition 1. Classic Definition of Service (Manual Service)

A **service** consists of actions performed by an entity on behalf of another.
It is an asset with an inherent value. The consumption of a service involves the transfer of value and (mostly) the generation of cost with the consequent need of its settlement.

The "Unique Characteristics of Services" are:

1. A service is intangible.
2. A service is heterogeneous.
3. The production of a service is inseparable from its consumption.
4. A service is perishable, cannot be inventoried.

Generally, a service is not simply a function, it is a function performed by a **Service Provider** on behalf of another entity, commonly the **Service Requester** or **Service Client**/Customer. The consumption of a service transfers a value from the provider to the customer. The definition of the service (Definition 1) also mentions the potential generation of costs, which has to be settled (mostly) by the consumer of the service (service could be a **Business Service**). The costs of a service are an example of **non-functional properties** (**nfp**) connected with a service. Reliability, availability, and Quality-of-Service are further examples of non-functional properties of services. A service with a bank is likely split into separate transactions, which have to be controlled. The transactions could be services on their own. In this way, a service can be composed on different levels of abstractions. This kind of abstraction of services leads to the distinction between Composite Services, Component Services, and Basic Services or Atomic Services.

Example 1 illustrates a simple **Technical Service** realising a telecommunication service, an audio communication using a gateway service. The technical service is a Composite Service with more than one Service Client. A telecommunication service can be described *as a packaged set of capabilities for exchange information over distances, perceived by human end user* [Bet2008]. On the one hand, a Technical Service can be treated as a Business Service, since it has to be paid for the use of a telephone. Therefore, both Service Clients have a client-server relationship with the service of the communication facility (cf. Fig. 1). On another hand, the Technical Service consists of several Component Services of communication facilities, in order to realise the technical function. The relationship between these Component Services of a Technical Service can be called non-directed or peer-to-peer relationship, because both communication facilities are equally involved. A Service Provider of a telecommunication service will only advertise a telephone service, but not the technical Component Services for its realisation. The different view on a service is reflected in the Service Descriptions. Business Services distinguish between Business-to-Customer Service Composition (B2C) and Business-to-Business Service Composition (B2B).

Characteristic	Service Category			
	Physical Acts to Customers' Bodies (e.g. passenger transport, restaurant service)	Physical Acts to Owned Objects (e.g. freight transport, repair, health care)	Non-physical Acts to Customers' Minds (e.g. education, entertainment)	Processing of Information and Communication (e.g. cloud, Web 2.0, CSCW)
Intangibility	Performance is ephemeral, but experience may be highly tangible and even result in physical changes.	Misleading point: performance is ephemeral but may physically transform possession in tangible ways.	Yes.	Yes.
Heterogeneity	Yes. Often hard to standardize because of direct labour and customer involvement	Numerous exceptions. Service can often be standardised	Numerous exceptions. Services can often be standardised	Numerous exceptions. Services can often be standardised
Inseparability of production and consumption	Yes. During production	No. Customer is usually absent during production.	Only when performance is delivered "live".	Many exceptions. Customers are often absent.
Perishability, i.e. service cannot be inventoried after production.	Yes	Yes	Numerous exceptions. Performance can often be stored in electronic or printed form.	Many exceptions. Performance can often be stored in electronic or printed form.

Table 1 - Applicability of "Unique Characteristics of Services"

It is up to the Service Provider, if he advertises only Concrete Services or an Abstract Service (another kind of abstraction). For example, a travel agency might use Abstract Services, in order to advertise journeys. The concrete journey is a result of negotiation between Service Requester and Service Provider. The orchestration of Composite Services in Component Services and the coordination of the execution of the functions of the Concrete Services are common challenges of Service Providers and Service Clients. These challenges led to an inherent complexity of the Service Description, which is disclosed to the different service users according given policies, e.g. the orchestration may be transparent for a Service Client. The used policies can be used for a categorization of services. Besides the inherent complexity, services can be categorized through further aspects, which will be discussed in section 2.3.

Example 1 **A Communication Service**

Fig. 1 shows the telecommunication between a Service Client A and a Service Client B, which might be requested through their avatars in the working zone. A is using a mobile and B an IP-phone. In order to establish the communication connection between the two Service Clients, a gateway or gatekeeper might be necessary for the conversion of signals and protocols consistent with different telecommunication standards. The standardised telecommunication protocol H.323 [IEC2007] allows the voice over IP communication.

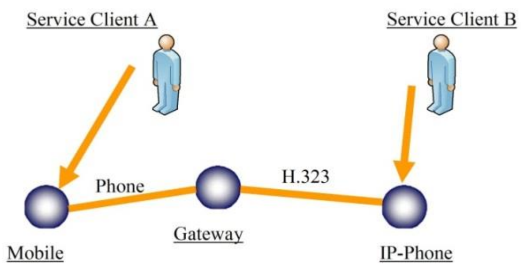


Fig. 1 - Communication Service

2.2 Electronic Service and Web Service

Early on, IT researchers and practitioners have commonly agreed about the importance and potential of e-services in Distributed Object Computing (DOC) or Open Distributed Processing (ODP) [ISOITU1996; ITUISO1997b; Joy2005]. An e-service was defined as an abstract, electronically processable notion that must be implemented by a concrete agent (Definition 2), which can be a concrete application, hardware, or a physical agent (e.g. Service Provider, actor in case of a manual service). A service in this context [ITUISO1997a; ODP 2004] is for instance the electronic exchange of information ("When goes the next train?"), the supply of computing power (e.g. grid, cloud), the access of a manual service (e.g. flight, hair-cutting), the offering of remote applications (remote procedure call (RPC), RMI, CORBA, CGI, servlets), or the access of technical facilities (JINI, ODBC, JDBC [Fra2007; OraSUN2005; WalArn2001]). In e-business, the necessary integration of the (distributed) resources of services with their application logic and presentation was addressed through Distributed Information Systems (DIS).

Definition 2. Electronic service (e-service)

An **e-service** is an abstract, electronically processable notion that must be implemented by a concrete agent. Ideally, the implementation and location of the agent is independent of the notion. The agent can be a concrete piece of software, hardware, or a physical agent that sends and receives messages.

Service-Oriented Computing (**SOC**) introduced the service as a new software paradigm similar to an object. Due to the loose coupling, Software Services can be easier reused than objects or modules, although their granularity is rather coarse. While initially services were meant to be a simple object access protocol, the provision of functionality independent of its implementation technology and execution location gained importance. SOC has to deal with two challenges: the (extended) **integration challenge** and the **semantic challenge**. The former is the challenge to integrate multiple independent and heterogeneous data repositories, processes, and applications in a dynamic system environment. The later tries to avoid semantic mismatches and to overcome the heterogeneity of data, in order to have a more reliable Service Discovery (cf. [Syc2010; VetLen2005]).

Definition 3. Web Service

A **Web Service** is a standardised electronic service [Ber2003], which uses URIs to identify resources and is based on XML. It is characterized through a provided set of functionality (e.g. in WSDL - Web Service Description Language, cf. [FarLau2007a]) and Internet based message protocols like SOAP (definition is based on [GudHad et al.2007]).

The standardizations of a Web Service (Definition 3) eased the integration of applications. New functionality is created from existing building blocks and communication is enabled between various elements. In contrast to the former approaches that addressed (at least partially) similar goals, such as CORBA or Multi-Agent Systems (MAS), the approach of Web Services takes advantage of simple, open, platform-independent protocols based on accepted Internet standards. WS techniques and standards allow the encapsulation of existing code and the integration of applications.

Berners-Lee [Ber2003] describes Web Services and their standards as belonging to the information space based on URI links and namespaces with an unambiguous identity. UDDI (Universal Description Discovery and Integration) [OAS2005] realises the name and directory server, the Discovery Agency of Web Services. Web Services allow the creation of self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web Services define functions that can be anything from simple requests to complicated business processes. Once a Web Service is published, applications can discover and invoke the service [W3C2006].

2.3 Aspects of services

Since the "Unique Characteristics of Services" do not hold for electronic services in all cases (e.g. the delivery and reading of an e-mail separates the production and consumption of a service), the next sections discuss four aspects of services specially introduced in this thesis, in order to achieve a classification of services. Through this classification, services can be categorized. Examples for criteria of this classification are shown in Table 3:

(1) Different views on a service,

(2) the non-functional attributes/properties (nfp) of a service,

(3) the inherent Complexity of a service, and

(4) The different phases of the life cycle of a service.

2.3.1 1st aspect of services: Different views on a service

The idea and task of a service differs with its application domain. For instance, a service in a technical domain (Technical Service) can be distinguished from a service in a mainly business oriented domain (Business Service). A Technical Service (e.g. Fig. 1) can have peer-to-peer Service Compositions and several Service Clients. A Business Service can have alternating client-server bindings.

Software Engineering has its own views on services. It looks at Software Services as a distribution model [ElfLay2002] and as a software paradigm [SaeJaf2005]. As a distribution model, the applying software can subscribe to the whole functionality of a software service, or it restricts its subscription to parts of the functionality (e.g. Cloud Computing). So-called autonomous services can be used as general software modules, which are not stateless or fixed to certain resources. Depending on the Service Provider the separations between autonomous agents (Fig. 6) and autonomous services disappear.

A requester of a service (Service Requester) has another view on a service than the provider of a service (Service Provider). Business Services distinguish between interfaces for Business-to-Customer (B2C) relationship and Business-to-Business (B2B) relationship. It is a similar case with Technical Services. Service Traders are acting with their assigned policies for Service Discovery and Service Composition. In their construction of a semantically enhanced environment for Semantic Web Services, Medjahed and Bouguettaya show that the community is another possible perspective for describing services and their framework [MedBou2005]. The user roles of the Service Client or the Service Requester inside of the community influence the reception of a service. A manager will have another access on a service than a worker. For instance, a manager might see more confidential data.

2.3.2 2nd aspect of services: non-functional attributes

Non-functional attributes are considered to be constraints over the functionality of the service [O'EdHo2002]. The literature alternatively speaks of attributes, properties, or parameters. In the thesis, mostly the terms parameters and properties are used. Some non-functional properties (nfp) are simply annotations like contributor, coverage, creator, date, format, identifier, language, owner, publisher, rights, source, and version. Traditionally, Service Discovery of e-services works with the functional description. The annotating non-functional properties are not really used for the Service Discovery. This implies a general assumption of the traditional Service Discovery environments that it is not likely that anybody wants only services of a certain publisher. Nevertheless, this category of nfp can be quite useful for the selection of services (e.g. most recent offered services). Another category of nfp is more helpful for several phases of the Service life cycle (4th aspect of service): Service Discovery, negotiation, monitoring, and substitution of services at run-time. This second nfp category describes service criteria like location, time, availability, obligation, price, payment, discounts, rights, trust, Quality-of-Service (QoS), security, intellectual property, rewards, provider, reliability, robustness, scalability, performance, transactional, and channels (usually a channel describes the way how a service is delivered).

For instance in Example 1, it would be unacceptable for Service Discovery, if the IP-phone was not accessible for spatial or security reasons, non-functional properties like availability of a service have to be considered, in order to ensure that the Component Services can be deployed. The dealing with non-functional properties like Service Quality, Settlement, or Warranty can involve new Component Services. Therefore, the support of non-functional properties increases the inherent complexity of a service (3rd aspect of service).

The monitoring of Service Quality may lead to a stop of the Service Delivery, when the detected values fall under a given threshold. A possible reaction to an interruption is the Service Substitution. In the worst case of substitution, the phases of the life cycle have to be repeated, in order to discover and deliver a new (component) service.

In the field of computer networking and other packet-switched telecommunication networks, the traffic engineering term Quality-of-Service (QoS) refers to resource reservation control mechanisms rather than the achieved service quality [EvaFil2007; Mar2007; Xia2008]. Quality-of-Service is the ability to adapt priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality-of-Service guarantees are important if the network capacity is insufficient. The standards organisation FIPA specified a Quality-of-Service Ontology [Dal2002b].

2.3.3 3rd aspect of services: The inherent complexity

The inherent complexity of a service is created through Service Composition and Service Coordination, i.e. that a service can consist of several Component Services, which have to coordinate their message exchange in order to achieve the goal state of the service. The process of Service Composition is recursive, since a Composite Service can be a Component Service of another Composite Service. The terms of **orchestration** and **choreography** are discussed in section 3.5. The choreography describes the behavioural aspect of services that means the possible message exchange sequences, which are called a process. Different kinds of Service Composition are distinguished depending besides others on the matching methods. The Service Composition of WS with BPEL is often called process oriented, whereas the Service Composition of SWS with Service Capabilities is rather service-oriented. The distinction between Abstract Services and Concrete Services has to be mentioned in this context as belonging to this aspect of services.

2.3.4 4th aspect of services: The extended life cycle of services

In Software Engineering, a Software Development Process, also addressed as a Software Development life cycle (SDLC), is a well-known structure imposed onto the development of a software product. As the software is often part of greater system environment, a software life cycle can be considered as a subset of a systems development life cycle. There are several models (e.g. Waterfall Model, Spiral Model, or Rapid Application Development) for processes. Each model describes a variety of tasks or activities that take place during the process. Typical activities in this context are Planning, Implementation, Testing, and Documentation. Some people consider a life cycle model as a more general term and a software development process as a more specific term. An international standard for software life cycle processes is ISO/IEC 12207[2]. It aims to be the standard that defines all the tasks required for developing and maintaining software.

Objects, software agents, and e-services can be software paradigms for the construction of a software product. They are the main entities of the activities during the mentioned process, and they have to be deployed and maintained when the software product gets alive or executional in the system environment. A manual service can directly be seen as a part of the system environment. With the introduction of the e-service, a service as a paradigm (including manual services) is also described through meta-information for the Service Discovery and Composition on various levels of abstraction. The design and application of such meta-information leads to a new, extended life cycle for Service Discovery, which is specific for the service paradigm. The fourth aspect of services considers this extended life cycle of services from two perspectives. According to the Service-oriented Architecture, a discovered service will be finally deployed by at least one Service Provider. The deployment phase can be seen as the connection between the Service Discovery life cycle and the software/manual (executional) life cycle of the service. In the thesis, the Service Discovery life cycle is addressed with the term (extended) life cycle.

On the one hand, the service life cycle can be described in several processes from the perspective of the Service Provider (Service Definition, Property Provision, Service Delivery) as well as from the perspective of the Service Requester (Provider Discovery, Property Discovery, Service Call). The Property Discovery and Property Provision are negotiating processes for the refinement of the Service Properties. The result of the negotiations is a Service Level Agreement (SLA). Service Call and Service Delivery are corresponding processes for the Service Consumption phase.

On the other hand, the life cycle can also be described in six phases (Fig. 2): In Phase 1 (Service Design), the Service Provider uses a given Service Description language, in order to describe the interface of the offered services (e.g. WSDL for Web Services). The Service Design is followed by the Service Discovery consisting of three phases: Service Trading, Service Matching, and Service Ranking/Checking. The SOA enables the publishing of the Service Definitions in Phase 2, the Service Trading. Service Trading can be done actively through Service Traders or passively with a Service Registry (for instance UDDI). An active Service Trading can include Automatic Service Composition (i.e. the dynamic composition of a new service from the

[2] http://en.wikipedia.org/wiki/ISO/IEC_12207

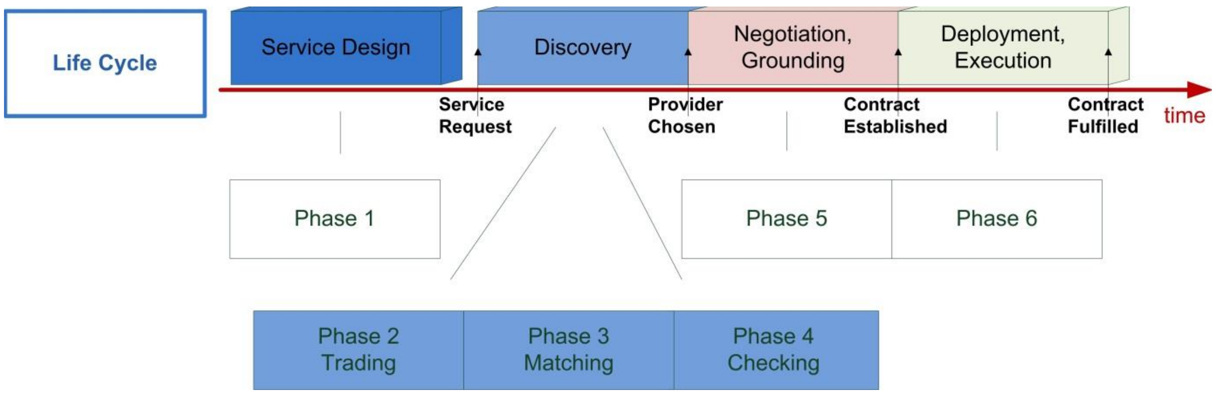


Fig. 2 - Phases of the extended life cycle of e-services

offered services). Active Service Trading is in principal independent from a concrete Service Request. The Service Matching (Phase 3) will take advantage of the Service Trading, when no fitting Service Provider is known (the process of Provider Discovery). The Service Request is the starting point of the Service Discovery and Service Matching process of Phase 3, which ends with the matching of Service Request and Service Offer in a successful case. In phase 4, the Service Ranking, the found Service Offers are checked and eventually ranked. Service Negotiation (Phase 5) is necessary for the refinement of Service Properties; it comprises the earlier mentioned processes of Property Discovery and Property Provision. During Service Grounding, system specific constraints for the Service Delivery (e.g. resource management) as well as coordination issues are solved. Possibly, the phases 3 to 5 must be repeated, if (component) services do not fulfil the resulting constraints of Service Negotiation/Grounding. In the Service Deployment (Phase 6), the Component Services are scheduled and deployed. The Service Execution might observe the Service Quality. In case of non-fulfilment of guaranteed QoS properties, earlier phases have to be repeated. A feedback for learning and keeping up the Service Quality might be provided at the end of phase 6.

Service Traders or Service Registries provide a vital aspect of services Discovery in SOA. However, already Vasudevan [Vas1998] introduced a proposal for taxonomy of traders (Table 2). Service Discovery can be based on a centralized or distributed Service Registry (cf. [RamHol et al.2009]). One or several, federated Service Traders can be involved in Service Discovery. Peer-to-Peer-Networks realise a federated Service Trading. A Service Broker is a Service Trader, which enacts as the discovered service. Web Service standards does not define a trading, but a central registry supporting the keyword of Service Offers: The Universal Description, Discovery and Integration (UDDI) [OAS2005; Org2004] specifications define a registry service for Web Services and for other electronic and non-electronic services. The UDDI consortium has released guidelines on how to use WSDL to describe service interfaces and store them in UDDI registries [CuEhRo2002].

Before the emergence of Web Services, ODP-Trading [Bea1997; Int1995] drafted a standard for trading; this draft inspired several trading approaches in the nineties (e.g. ,[JacMud1996; MarMer et al.2001; PudMar et al.1995; VoBeIa1995]). These traders already included QoS attributes [Kos1999] and semantic enhancements with ontology based IOPE Capability Descriptions [Ber2003; Klu2000b; Klu2000a].

In Table 3 - Examples of criteria for the four aspects of services, possible criteria for the classification of services through the introduced aspects are listed. For instance a service could belong to the domain of Cloud Computing and is described as SaaS from the perspective of the Service Provider (1st aspect). The service can be linked with distinct criteria of reliability, cost and settlement scheme (2nd aspect). The orchestration of the service might be completely transparent for the user, due to company policies (3rd aspect). The third aspect also distinguishes between Abstract and Concrete Services. Finally, the Service Description can be proprietarily adapted to the specific Cloud Services of a given Service Provider, who also offers an own Trading Environment for the available Cloud Services. At last, the Service Provider demands a distinct protocol for the handling of his services. In this way, the Service Provider generated several criteria belonging to the 4th aspect of services (phase 1, phase 2, and phase 5), which led to the so-called "Lock-in-Effect" (cf. section 3.6).

Criterion	Description
Knowledge Representation	Advertising of Service Offer to the trader (push scheme, pull scheme) Functional, semantic, adaptive (gained at runtime) Hierarchical or flat name/directory space (e.g. X.500 hierarchical, URI flat)
Matching Heuristic	Property-based or Artificial Intelligence (AI) concept-based
Service Invocation	Trader can be a matchmaker or broker (enactor)
Federation Approach	Single (monolithic) trader or cooperating (federated) traders.

Table 2 - Taxonomy of trading

2.4 Summary

The chapter showed that a commonly accepted definition of services does not exist, because the idea of a service changes with the domain and the perspective of the user. Alternatively, four aspects of services were discussed, which could be used for the classification of services.

Two challenges of SOC were pointed out: (1) the integration and (2) the semantic challenge. The former comes up at the integration of applications of different Service Providers, since various resources and actors have to be included, incompatible data schemata need mediation, as well as communication protocols, i.e. the choreography, must be harmonised. The semantic challenge looks for distinct informal descriptions of service functionality.

The integration challenge was tackled with the Web Service standards. Web Services define communication protocols, describe the functional interfaces, and allow a Service Discovery based on structure, syntax, and vocabulary. Semantic Web Service (SWS) extends the used vocabulary of the functionality description with semantics and pragmatics. SWS techniques ease the Autonomic Service Discovery, because they are an answer to the semantic challenge. The next chapters have a closer look at the integration challenge, the semantic challenge, and the Autonomic Service Discovery.

Aspect of Services	Examples
1.) View on service	• Domain: e.g. CSCW, software engineering, supply chain • Role in SOC: Service Provider, Service Requester, Service Client • Role of user: e.g. Administrator, Travel Agent, Researcher • Kind of cloud service: IaaS, PaaS, or SaaS
2.) Non-functional Aspect	e.g. QoS, trust, reliability, availability, user preferences
3.) Inherent Complexity,	• orchestration, choreography, coordination, transaction • Abstract Service vs. Concrete Service • Workflow Management Systems (WfMs) • Business Process Tools (cf. Fig. 12) • company policy, law, political correctness
4.) Life cycle of service	Phase 1 - Service Design • Functional view – syntactical or semantically enhanced • Service Request specification – requester view • Mediation – data, process, and protocol level • Service Composition on service and/or process level • non-functional properties – keywords or concepts Phase 2 – Service Trading • The kind of trading – Matchmaker, Broker, or P2P • Registry federation – central, distributed, decentral • Preferences, Learning, Trust • Trading Policy – keywords only, concepts, or Data Mining Phase 3 – Service Matching • Property based or enhanced concepts (e.g. IR in OWL-MX as an enhanced concept) • Functional Matching • only input/output parameters or complete IOPE • Intersection, plugin, subsumes, equivalence (exact) • Matching of Abstract Services or Concrete Services Phase 4 – Service Checking/Ranking • Preferences • Constraints • Prioritisation of Service Providers • Availability of Resources, Reservation Phase 5 – Service Negotiation and Service Grounding • Negotiation of further Service Properties • Agreeing on Concrete Services if Abstract Services matched • Tools for Service Composition on process level • Agreement on Web Service standards (e.g. WS-I) Phase 6 - Deployment and Execution • Deployment Environment • Monitoring QoS • Robustness • Integration into the application environment (e.g. VPO) • Feedback (especially for learning of Preferences)

Table 3 - Examples of criteria for the four aspects of services

3 The integration challenge

The integration challenge can be split in two tasks that have to be solved:

- The first task is the integration of resources and actors into the environment of an application, whereby the resources and actors are possibly located in a dynamic and distributed environment.
- The second task is the integration of applications, in order to take advantage of existing solutions.

In e-business, the first task was tackled with (Distributed) Information Systems (DIS) (cf. [AloCas et al.2004 section 1]), which offered an application as an e-service. Since the environment of a DIS was mostly under control of one enterprise, the application integration could often also be realized in the application logic layer of the DIS, because the policies and security rules of the controlling enterprise could be adapted accordingly.

The challenge of the application integration increased, when the applications were running on separated DIS environments controlled by different enterprises. The DIS concept had to be extended through the concept of Enterprise Application Integration (EAI), in order to allow application integration over several enterprises. SOC with the composition of services came alive and it has moved mainstream with the introduction of the standards of Web Services for the last decade, as the realisation of DIS with EAI extensions became realistic.

The **Service-oriented Architecture** (**SOA**) allows the dealing with dynamic environments. The Service Provider can advertise a service with a Discovery Agency. Through this Discovery Agency, a Service Requester can discover advertised services, which have a Service Description fulfilling the Service Request (Fig. 3). From 2007 onwards, WS technology initiated Cloud Computing, which extended the SaaS concept in the direction of a general e-service (cf. Definition 2).

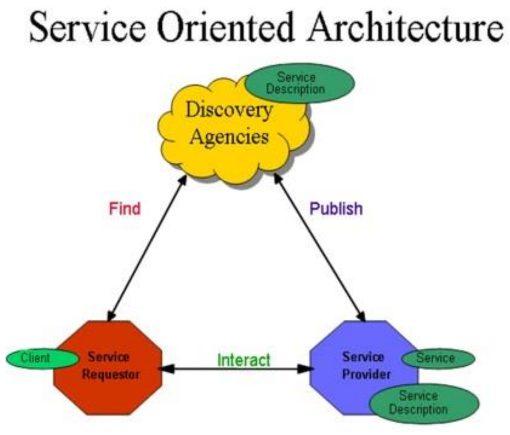


Fig. 3 - SOA

In this chapter, the integration challenge is illustrated. DIS middleware approaches are discussed and the standards of Web Services introduced. Finally the paradigm of Cloud Computing is presented. In the subsequent thesis, DIS and EAI are used synonymously for distributed information systems, which encompass the EAI concept of application integration.

3.1 Motivation and Evolution of DIS middleware

In the domain of Computer Supported Cooperative Work (CSCW), the first task of the integration challenge became for instance apparent in the supporting of a world-wide dispersed project-team (Example 2). A Virtual Project Office (VPO) showed with avatars the currently working project members. The interface integrated various services: information services, business services, and technical services. Additional context information was linked with so-called working zones. The availability of the services and the number of people currently working at the project changed constantly. Additionally, the team members were not bound to specific locations, since they could access the VPO interface through the Internet. In order to establish transparently a communication service, the currently accessible communication facilities of the intended Service Clients have to be discovered and deployed automatically. Without WS technology, VPO became a non-modular approach. The middleware was realised with a Multi-Agent System (MAS) and resources were accessed through technologies like JINI or ODBC. The transparent access of resources already reflects the utilisation idea of Cloud Computing. However, Cloud Computing relies on the service paradigm, in order to have a standardised interface and on-demand accounting (cf. section 3.6).

JINI[3] [WeiBel2002] offered an early, Java based solution of the integration challenge in the domain of technical services. Technical facilities became accessible with a unified interface as technical services (e.g. in VPO). A Service Provider could advertise a JINI service through a lookup service (LUS) with a Service Registrar object to the service. The LUS returned a Java proxy, specifying how to connect directly to the service. The availability of services was addressed in JINI through the concept of leasing. When a service registered with a LUS, a lease of a specified duration was granted. Leases had to be periodically renewed. In this way, JINI addressed availability and the Service-oriented Architecture (SOA) (Fig. 3) on the level of resources. The disadvantage of JINI was its proprietary standard depending on Java.

The classical three-layer design of an Information System integrates the Application Logic Layer as middleware into a Presentation Layer and a Resource Management Layer. Wrappers like JINI in the Resource Management Layer allowed the reaction on changing resource environments without the need of changing the application logic. The **service** (Definition 2) paradigm became the external presentation of the application and was introduced as a specific user interface.

Early DIS approaches evolved from the Remote Procedure Calling (RPC) concept of Open Distributed Processing (ODP). The RPC-based systems were soon extended with transaction control. The resulting TP Monitors were according to [AloCas et al.2004] long time the most known and stable technology of DIS middleware (IBM CICS, MS MTS, BEA Tuxedo). ODP

[3] Jini.org, 24.11.2010

standardised services as interfaces and they considered availability (Service Template/Offer) and SOA like concepts in its standardization [ISOITU1996] similar to the discussed concepts of JINI.

The middleware technology changed with the introduction of object oriented programming and Distributed Object Computing (DOC). Object Brokers and Object Monitors (the latter were Object Brokers extended with approved TP Monitor techniques) took over. Remote Method Invocation (RMI) allowed the RPC access in Java. Object repositories like Java Beans realised the offering of business algorithms independent of concrete applications.

Example 2 A Virtual Project Office (VPO)

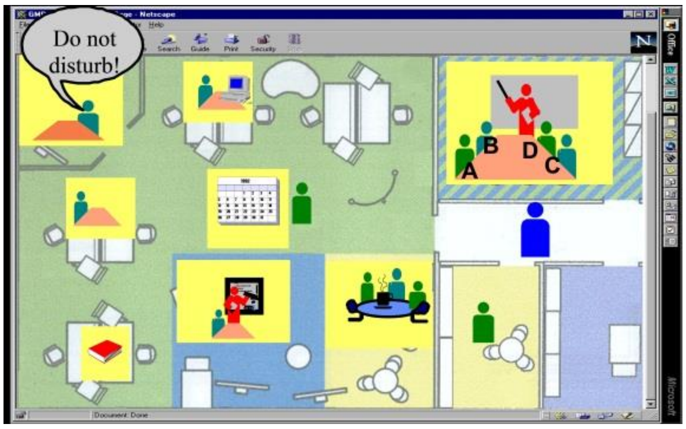


Fig. 4 - VPO with Working Zones

It was the goal of the project Virtual Project Office (VPO) [BahBur et al.1999; ReiBah et al.2000] to support the project work within a world-wide dispersed team. The virtual working area was designed like a physical working area. It offered different working zones. The principal VPO in Fig. 4 consists of seven zones: a conference zone, a normal working zone with some desks, a special working zone for undisturbed work, a presentation zone, an informal communication zone, an archive or library zone, and a general service zone. In the virtual working area of VPO, a team member was shown as a personalised avatar. The avatar of a person could only be in one VPO and one concrete zone. The symbolic representations stood for specific services of information requisition, communication and co-operation. For instance, when the avatar of a team member entered the conference zone, then he became automatically part of an on-going online conference. In a non-disturbed zone, only asynchronous communication was generally possible. The positioning of an avatar in a certain zone also signalled the kind of conversation wanted. The challenge of the VPO environment was the transparent establishment of the services. For instance a technical communication service could only be established with the currently available communication facilities of the team members, who wished to become Service Clients. Thus, some non-functional attributes like accessibility, security, and resource management were involved. The services of the communication facilities had to be composed and coordinated. For this purpose, the Service Discovery mechanism had to observe the technical compatibility as well as the current environment.

The most popular object brokers were those based on Common Object Request Broker Architecture (CORBA), defined and standardised by the Object Management Group (OMG) [OMG2008]. CORBA standardised the mapping of the Interface Description Language (IDL) to different programming languages. Object Brokers also provided an encapsulation for location independence through Object Request Brokers (ORB), which advertised the access to the server objects and could act as brokers for the object access through interoperability with other ORBs using the General Inter-ORB Protocol (GIOP). The system architecture of CORBA provided a set of CORBA services (e.g. transaction, events, security) for the object access and a set of so-called CORBA facilities, which were higher-level services needed for the support on application level. The CORBA facilities were distinguished between vertical facilities like supply chain support and horizontal facilities like information management.

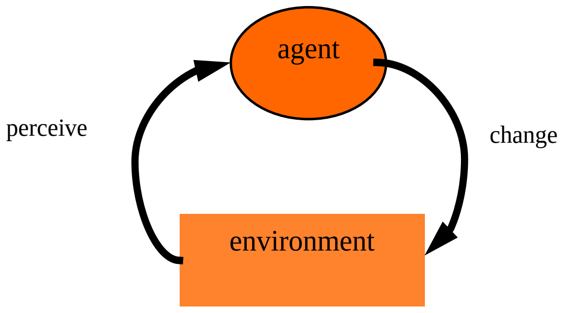
However, EAI realised through CORBA middleware turned often out as incompatible, since the implementation was not included in the standardization. The goal of an alternative approach, Distributed Computing Environment (DCE) [Hou1996], provided by the Open Software Foundation (OSF), was the provision of a standard implementation, which the vendors could then use and extend as needed for their own products. By using the same basic implementation of RPC, the hope was that the resulting products would be compatible [AloCas et al.2004 p. 43]. Even when DCE made a point and was used in some EAI middleware, it was rejected as too restrictive by most developers.

A new class of middleware, the Message-Oriented Middleware (MOM) allowed asynchronous bindings between client and server and made peer-to-peer bindings possible. Asynchronous RPC was introduced and the TP Monitors were extended with persistent message queuing (MQ) supporting local and remote queues (e.g. Microsoft Message Queuing, Java Message Service (JMS)). Message Brokers, which are part of many EAI middleware architectures, are a specific kind of Message-Oriented Middleware that has the capability of transforming and filtering messages as they move through the queues. They can also dynamically select message recipients based on the message content. In terms of basic infrastructure, Message Brokers are just queuing systems. The only difference is that application logic can be attached to the queues. This feature allows designers to implement more sophisticated interactions in an asynchronous manner. For instance DIS environments with Enterprise Application Integration (EAI) can dynamically deal with the integration of applications of a known type without a change of the application logic (Tibco ActiveEnterprise, BEA Web Logic Integration, WebMethods Enterprise, Microsoft BizTalk, and IBM WebSqhereMQ).

The evolution of the Web allowed standardised access of services through CGI processes or servlet threads on the server side and applets on the client side with HTTP tunnelling firewalls. An Application Server as DIS middleware unified the presentation of services over the Web including the provision of Web Services. The .NET and JavaEE initiatives offer the main development environments for Web based DIS.

3.2 Delimitation of Software Paradigms

A decade ago, a new software paradigm became popular: the Autonomous Software Agent [Woo2000] (Fig. 6). A Software Agent had the features of autonomy, re-activeness, pro-activeness, and communicativeness. A re-active Software Agent perceives and changes the environment (Fig. 5). The pro-activeness of Software Agents allows the integration of logic and the concepts of Distributed Artificial Intelligence (DAI), which allowed the creation of intelligent agents (e.g. BDI-agents [Woo2000]) for simulation, and distributed learning. Due to their communicativeness, several Software Agents can work pro-actively together in a Multi-Agent System (MAS) [HuhSte2000]. A group agent can work as an integration of several agents. For instance the virtual project offices and their working zones in Example 2 could have each an associated group agent, which controls the access of its working zone or VPO, respectively (Zone Agent (ZA) and Team Agent (TA) in Fig. 7). A ZA could recognize the request of a communication between team members. An initiated negotiation between the Personal Agents (PA) of the Service Clients and the available Facility Services (FS), managing the communication facilities (CF), led to the deployment of the communication service.



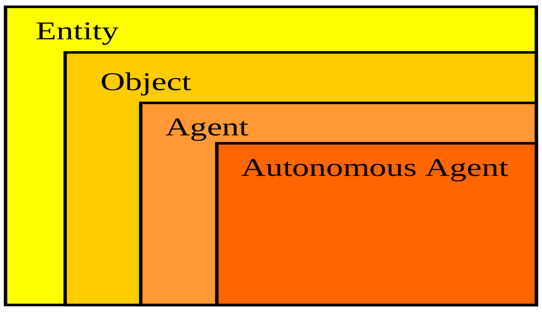


Fig. 5 - Agent Definition

Fig. 6 - Development of Software Paradigm

MAS middleware (e.g. Jade [Gri2010]) allows the creation and operation of MAS. Agents can be seen as carrying the vertical protocols of the application logic. The horizontal protocols are delivered by platforms or places. Autonomous agents can migrate from one place to another one, in order to take advantage of different protocols or services. Since standardization is the base of Application Integration, FIPA (Foundation for Intelligent Physical Agents) [Dal2011], the standards organization for agents and Multi-Agent systems was officially accepted by the IEEE as its eleventh standards committee on 8 June 2005. FIPA specifications represent a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they represent. Long time, the standards for agents and agent-based systems lacked integration in non-agent software engineering technologies. The association with IEEE was a not really successful attempt to overcome this lack.

JADE [BePoRi1999; Cai2009; Gri2010] of TILAB is a FIPA-compliant [Dal2005] MAS middleware. It allows the coordination of multiple FIPA-compliant agent platforms and their agents through the use of standard FIPA-ACL [Dal2002a]. Among other features it supports agent behaviours, asynchronous messaging, and multiple communication protocols, including RMI, CORBA, HTTP, and JMS. JADE provides mechanisms for resource discovery and several security features. The extension Wade [Cai2010] implements a workflow system on top of agents.

A MAS middleware, which is service-oriented but does not claim to be FIPA compliant, is JIAC (Java-based Intelligent Agent Componentware). JIAC V [DAI2008] is a Java-based agent architecture and framework that eases the development and the operation of large-scale, distributed applications and services. The framework supports the design, implementation, and deployment of software agent systems. The entire software development process, from conception to deployment of full software systems, is supported. JIAC allows a reusing of applications and services. The focal points of JIAC are distribution, scalability, adaptability and autonomy. The current version of JIAC incorporates ActiveMQ-based messaging, transparent distribution, service-based interaction as well as semantic service search and selection.

The distinction between objects, agents, and services is not absolute and no generally agreed definition of any entity exists. On the one hand, an agent offers a message interface, which can be accessed as a presentation for instance through a browser over Web. On the other hand, the implementation of a service could be interpreted as a non-autonomous agent, which reacts on requests (cf. Fig. 6). An e-service is an abstract notion that must be implemented by a concrete agent (Definition 2). This agent can be a piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of provided functionality. The realising agent of an e-service can be an autonomous software agent. In this way, (group) policies, planning, negotiating, and learning can be integrated in the distribution model of Software Services. In Software Engineering, services and agents are generally seen as coarser software components in comparison to distributed objects. They do not support

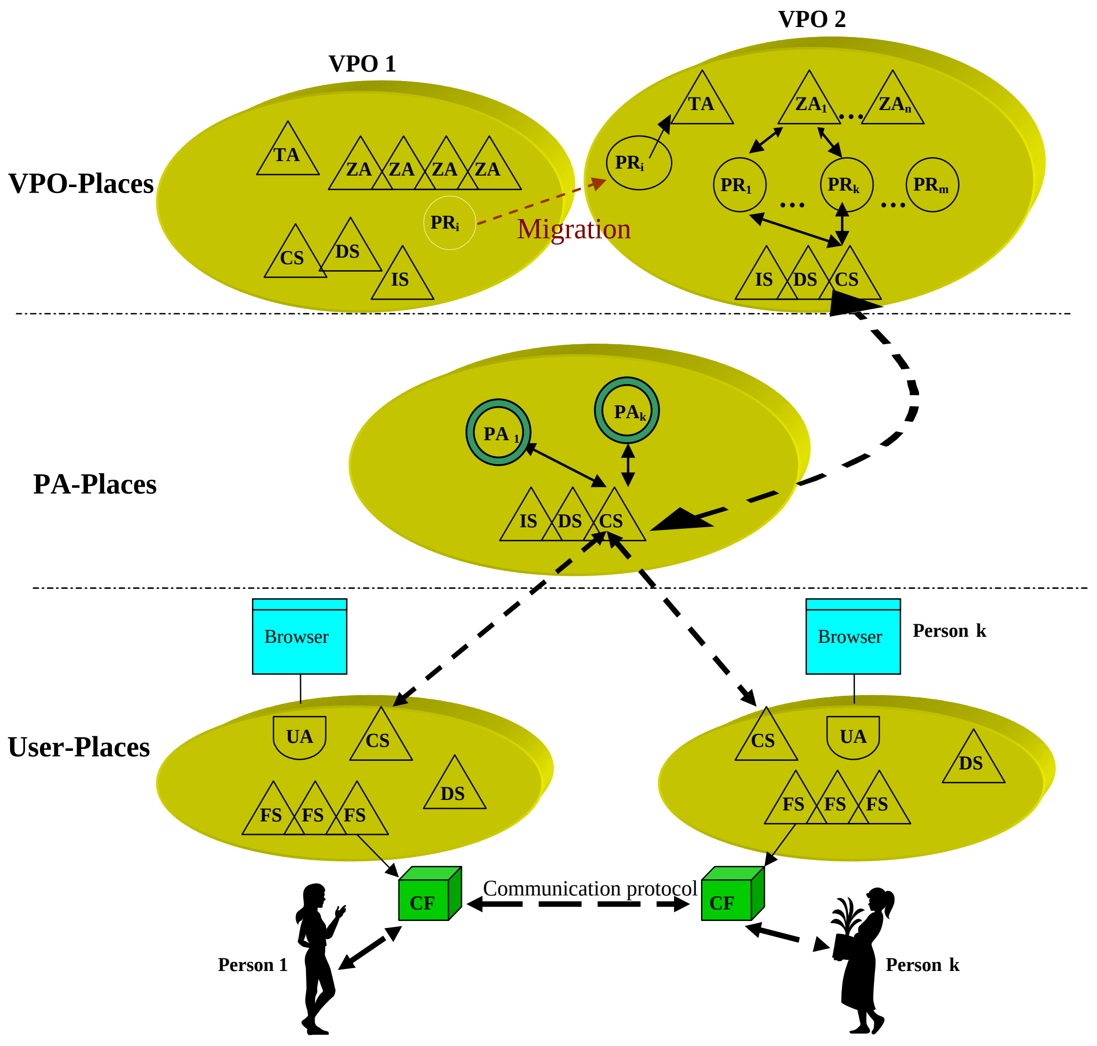


Fig. 7 - Principal Multi-Agent-System of VPO environment

inheritance and have in so far an easier usable interface, which is independent of the programming language used to implement the interface. The conceptual elements of objects like unique object identification, methods, class and instance variables, inheritance, and other concepts are not part of the Web Service model.

The introduced middleware approaches of a DIS including MAS middleware did not really work out due to three main reasons: (1) Complexity, (2) Lack of standardization, and (3) Political causes. An example for the latter was the mutual neglect of proprietary standards of concurrent companies. The complex and non-standardised communication protocols led to a difficult implementation, incompatibility and non-acceptance (e.g. MS DCOM was never completely compatible with SUN JavaEE due to proprietary standards).

In the past decade, the standardisation of Web Services (WS) [BooHaa et al.2004] was accepted as an answer to the integration challenge and an improvement for the realisation of DIS [AloCas et al.2004]. In the publication [DicWoo2005], Wooldridge and Dickinson compare Web Services and Software Agents. They argue that agents provide a distinctive additional capability in mediating user goals to determine service invocation. Huhns [Huh2006] sees that services themselves developing in the direction of software agents, and that services acting together can function as computational mechanisms in their own right. Thus, they significantly enhance the ability to model, and manage complex software systems. Service Discovery and Service Composition can take advantage of software agents. On the one hand, the software agents can enact the services, in order to find a feasible Service Coordination. On the other hand, in a horizontal architecture, they can play pro-active roles in the Service Composition, in order to adapt different policies for the composition, to enact an involved actor, or to sense the context of non-functional attributes.

In the next section, the standards of Web Services are considered. In the following sections, the perspective on the integration challenge is extended to Application Integration.

3.3 Web Service standards

Web Services solve the integration challenge through the use of standardised and open web technology, from URIs as the main addressing scheme over XML as the basic description language to the use of Internet protocols for message transport. Web Services are integrated through the exchange of XML messages as well as their creation is based on XML messages.

The Web Service standards are built bottom up, i.e. they begin with a simple transport protocol (SOAP) and Service Description (WSDL). The transport protocol SOAP [GudHad et al.2007] was first introduced as "Simple Object Access Protocol" supporting a RPC access. Web Services soon adapted the MOM approach and SOAP was renamed to "Service-oriented Architecture Protocol" (SOAP) or "XML Protocol" (XMLP). SOAP just defines a general pattern of how XML Web Service messages have to look like. Additionally, the standard outlines message exchange patterns and the encoding of XML type information. The message transport

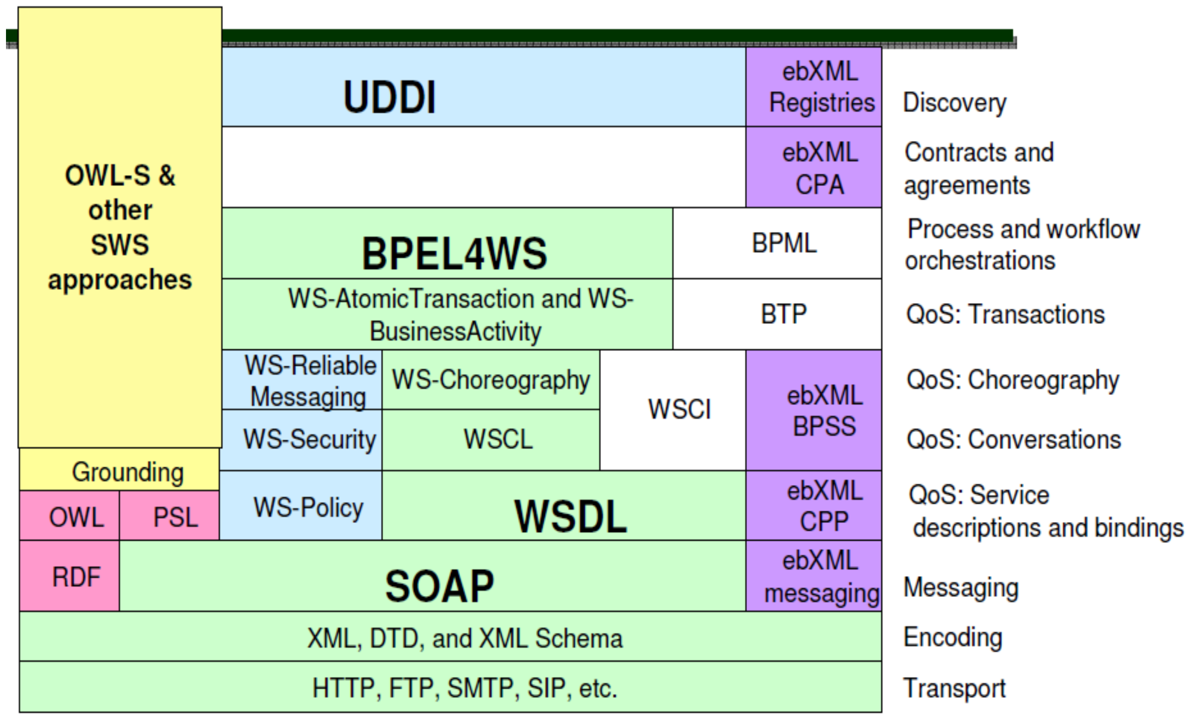


Fig. 8 - Web Service and ebXML standards

uses mostly HTTP. However since HTTP is a server-client binding, the transport layer can alternatively use an e-mail protocol like SMTP, or a Message Queuing System like JMS, in order to realize peer-to-peer or point-to-multipoint (MEP) bindings (cf. [StGrAb2007]). XML-RPC and REST (Representational State Transfer) protocols are two alternative transport protocols to SOAP. XML-RPC claims to be easier to use than SOAP for RPC bindings. REST is in principle a more restricted and strict SOAP protocol [BooHaa et al.2004]. "RESTful" applications use basic Web protocols like HTTP, whereas SOAP uses only a few commands of HTTP like HTTP POST.

Web Services Description Language (**WSDL**) is defined in XML and is used for the description of the Web Service interfaces. It describes how incoming and outgoing messages look like and where such messages are available (in terms of URI). It defines the methods, data encoding formats, and protocols supported by a service. Once such a definition exists and gets published, a Service Client can derive directly (e.g. with an XML parser) the outgoing messages for the service call. The Service Provider knows the possible incoming messages. Supporting tools allow the generation of binding code for the service and the accessing client application. WSDL offers message oriented and RPC oriented access.

The **Universal Description, Discovery and Integration** (**UDDI**) [Org2004] specifications define a registry service for Web Services with centralised repositories. A UDDI registry service can be implemented as a Web Service that manages information about Service Providers (White Pages), service classification (Yellow Pages) as well as service implementations, and service metadata (Green Pages, tModels). The UDDI V2.0 and 3.0 specifications have been approved as

OASIS Standards and are maintained by the OASIS UDDI Specification technical committee [OAS2005]. UDDI is mainly used for the storage of WSDL interfaces through so-called tModels [CuEhRo2002]. The tModels were introduced in UDDI, in order to store additional information in the repository. In the publication [AkkFar et al.2005], the World Wide Web Consortium (W3C) describes the storing of WSDL-S descriptions in UDDI registries.

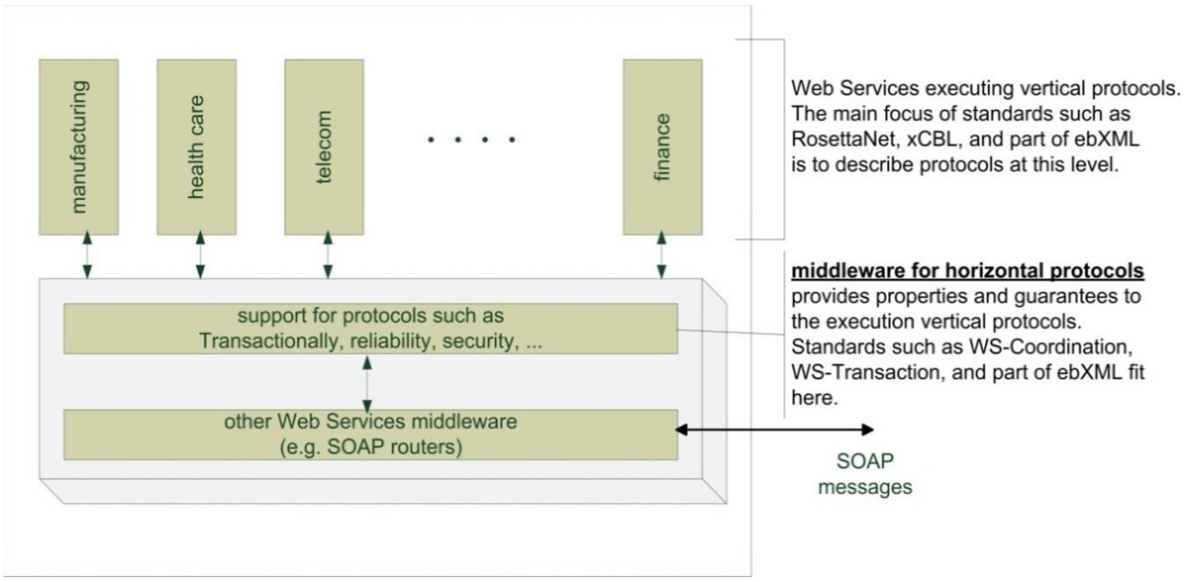

manufacturing
health care
telecom
finance
Web Services executing vertical protocols. The main focus of standards such as RosettaNet, xCBL, and part of ebXML is to describe protocols at this level.
support for protocols such as Transactionally, reliability, security, ...
other Web Services middleware (e.g. SOAP routers)
middleware for horizontal protocols provides properties and guarantees to the execution vertical protocols. Standards such as WS-Coordination, WS-Transaction, and part of ebXML fit here.
SOAP messages

Fig. 9 - Horizontal and vertical protocols (adapted from [AloCas et al.2004])

For the deployment of a Web Service application standards are supplemented. In Fig. 9, a basic architecture of the application logic of a Web Service agent is shown. The middleware layers hold the horizontal protocols and a set of vertical protocols realising the functions of the Web Service. The horizontal protocols do the basic and secure messaging, which is extended with protocols for transaction, reliability, and security. So-called **business processes** are standardised for the use as vertical protocols. They depend on the domain, the enterprise environment, and the different policies valid in the context (cf. section 3.4). The implementation of Web Service can be based on one of these vertical protocols. The middleware controls the given business process. Every implementation of a Web Service is based on basic horizontal protocols, like WS-Transaction, WS-Security, and WS-Addressing.

WS-Transaction (WS-TX) [OAS2009c] standards are relevant for the support of transactions, e.g. the transaction management of databases. Sub-standards are WS-Coordination [FeiJey2009], WS-AtomicTransaction [OAS2009a], and WS-BusinessActivity [OAS2009b]. WS-Coordination is a meta-protocol, which builds an extensible framework of providing protocols that coordinate the actions/operations of distributed applications/services. WS-Coordination describes how services can make use of predefined coordination contexts, in order to subscribe to a particular role in a collaborative activity. It can also be used independently from the other WS-Transaction standards (described in [AloCas et al.2004]). WS-Transaction provides a framework for incorporating transactional semantics into coordinated activities. The supported transactions can be simple (atomic transaction) or include complex business logic (business activity). The registered services can be freely distributed, since the propagation of activity is supported.

WS-Addressing [ChrFer et al.2004] proposes a protocol-neutral mechanism for specifying endpoint references of services within SOAP messages. WS-Security (Web Services Security, short WSS) [OAS2006b; OAS2006c] addresses the original security weakness of Web Services. The protocol specifies how integrity and confidentiality can be enforced on messages and allows the communication of various security token formats, such as SAML, Kerberos, and X.509. Its main focus is the use of XML Signature and XML Encryption, in order to provide end-to-end security on application level. WS-Trust [NadGoo et al.2007] extends the WS-Security specification with managed and signed Security Tokens for a trustful interaction of subjects of possibly different domains.

3.4 Service Composition and Service Coordination

Enterprises have different policies, servers, and data formats. The orchestration into Component Services might be hidden due to company policies. The Service Composition is determined through B2B or B2C relations. Fig. 11 shows a classic case: A supply chain is composed of several services provided by different companies with their proprietary policies and environments. In the supply chain, the interactions are implemented through the use of different (distributed) information systems. Companies maintain extensive customer, product, and supplier databases. The involved systems are heterogeneous: different operating systems, interfaces, and functionalities. They reside in different geographic locations. Each department is autonomously managed. It uses its systems to perform a variety of department-specific functions whose needs and goals are not necessarily aligned with those of the integrating application. Non-existing standards and the confidential character of the business policies (e.g. a company may not want to reveal its suppliers) made application integration involving several enterprises a difficult task.

Standardized interaction models for the Application Logic Layer helped to realise the concepts of Enterprise Application Integration (cf. [AloCas et al.2004]), i.e. the integration of applications running on middleware systems with heterogeneous enterprise environments. The service paradigm, introduced as an interface description, which does not comprises object concepts like inheritance, eased the loose coupling of applications of different enterprises.

The integration of applications extents the integration challenge to Service Composition and Service Coordination. In the Service Composition, Basic or Atomic Services become Component Services of a Composite Service, which itself can become a Component Service of another Composite Service. Different domains developed their own ideas of integration logic, i.e. the matching of compatible Component Services for the composition. An Atomic Service can be advertised as an Abstract Service by the Service Provider. The determination of the Concrete Services of a Composite Service is task of the Service Grounding. In the Service Deployment, the applications (or agents) of the Concrete Services have to be deployed, and the execution of these applications has to be coordinated.

The application integration logic in an EAI middleware involves business processes, the choreography of several applications, and the orchestration of an application in several sub-tasks. A **choreography**, which is also called a coordination protocol [AloCas et al.2004; BooHaa et al.2004] (in this thesis, coordination is a matter of deployment), is a model of the global sequence of operations, states, and conditions that control the interactions involved in the participating services. It is a multiparty (at least 2 parties) communication protocol, in which a party can adopt several roles. A choreography between two parties is often called a **conversation** ([Pre2007]). Example 4 shows a choreography with a (composite) Business Service. The order of the messages is constrained. The client sends an inquiry for a travel with insurance. He accepts the offer of Travel Agency 2 with receiving a bill and communicates consequently with the bank. The server or client role of a Component Service depends on the current business relationship.

Example 3 **Supply Chain - EAI**

> The responding to an RFQ (Request for Quotation) (Fig. 11) involves checking the availability of the product, the production schedule, and an extended checking with suppliers for delivery dates and prices for the required components. Processing the purchase order may involve interacting with a warehouse control system that indicates the current stock levels of the requested product and where it can be obtained. As part of the order fulfilment step, the purchase order may be forwarded to a manufacturing system. In this case possible additional steps are the purchase of components from suppliers, the arrangement for delivery dates, the schedule of the production and the testing. Finally, shipment and billing also require interactions with invoice databases.

Commonly, a choreography declares the constraints of the communication protocol globally. In the case of the supply chain (Fig. 11), it might clarify the communication protocol between customer, supplier and warehouse. Nevertheless, a global choreography can be transformed in a local choreography, which describes the exchange of messages from the viewpoint of one party complying with the constraints of the global choreography. A local choreography could for instance declare the constraints of the messages sent and received by the customer (for details cf. [AloCas et al.2004]). Messages between the supplier and warehouse are transparent for the customer, but the choreography might allow that he gets a confirmation message from the warehouse, although he sent the related order message to the supplier in the first place. Such local choreographies can be used for the matching of Component Services (e.g. [MarPim2010]), in order to achieve a Service Composition on process level.

Example 4 **Booking of a Travel**

> When a person wants to book a travel, he makes an inquiry at a Travel Agency. If the inquiry comes out with a satisfying result, he will make a booking. Finally, the chosen travel agency requests a “Billing” service. The person serves this request with “consuming” a paying service of a bank.

The message interaction prescribed by a choreography results in the completion of some useful common business goal. The **orchestration** specifies the order, in which concrete services are invoked, and the conditions under which a certain service may or may not be invoked; in particular, it defines the sequence and conditions, in which one Web Service invokes other Web Services in order to realize some useful local business goal. Since the introduction of message oriented DIS, the invocation of a service has been connected with a message and in this way it became part of a communication. Therefore, the terms (local) choreography, coordinati on, and orchestration have no generally excepted delimitation in the literature.

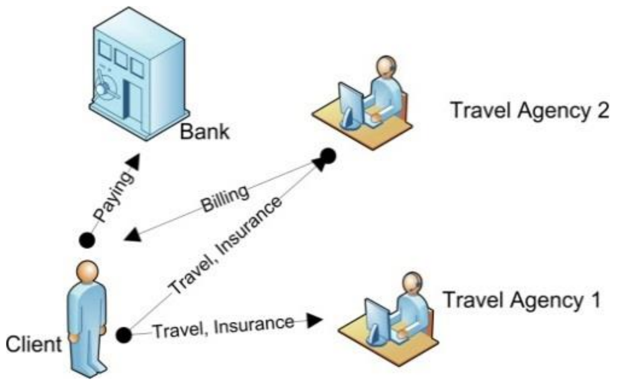


Fig. 10 - Business Service

In e-business, **business processes** are closely related to choreography and orchestration. Business processes of a supply chain are for example purchase orders, price negotiations, shipping management, and request for quotations (RFQ)). In duality with the distinction between DIS and EAI, private business processes and public business processes (also called collaborative

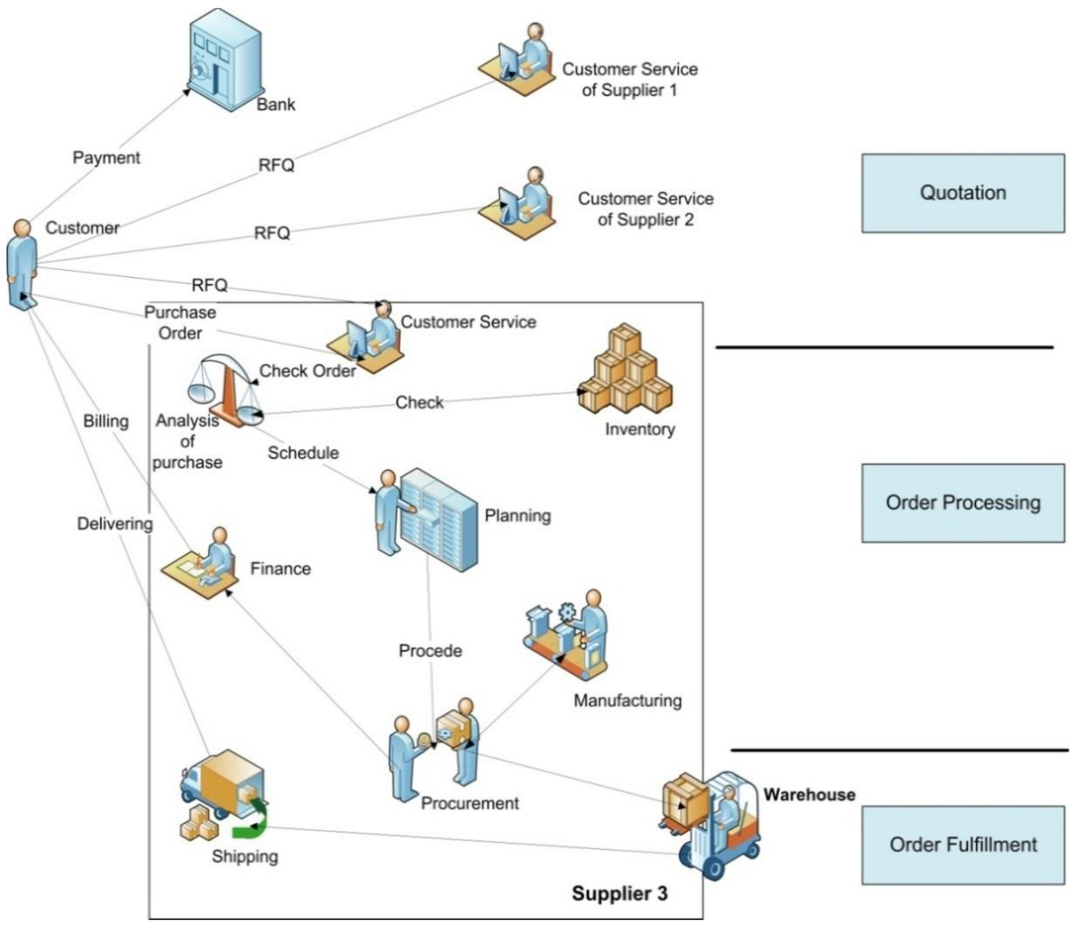


Fig. 11 - Supply Chain (typical EAI application)

business processes) are distinguished. Business processes are declared as private, if they are internal to an enterprise. Workflow Management Systems (WfMs) are seen as the "executing" environments of application integration for business processes of a confined domain (e.g. within a company) having a centralized enactment engine, i.e. they are often realised in the application logic layer of a DIS. Business Process Management Systems (BPMs) claim to follow a more general EAI approach for the application integration of public business processes.

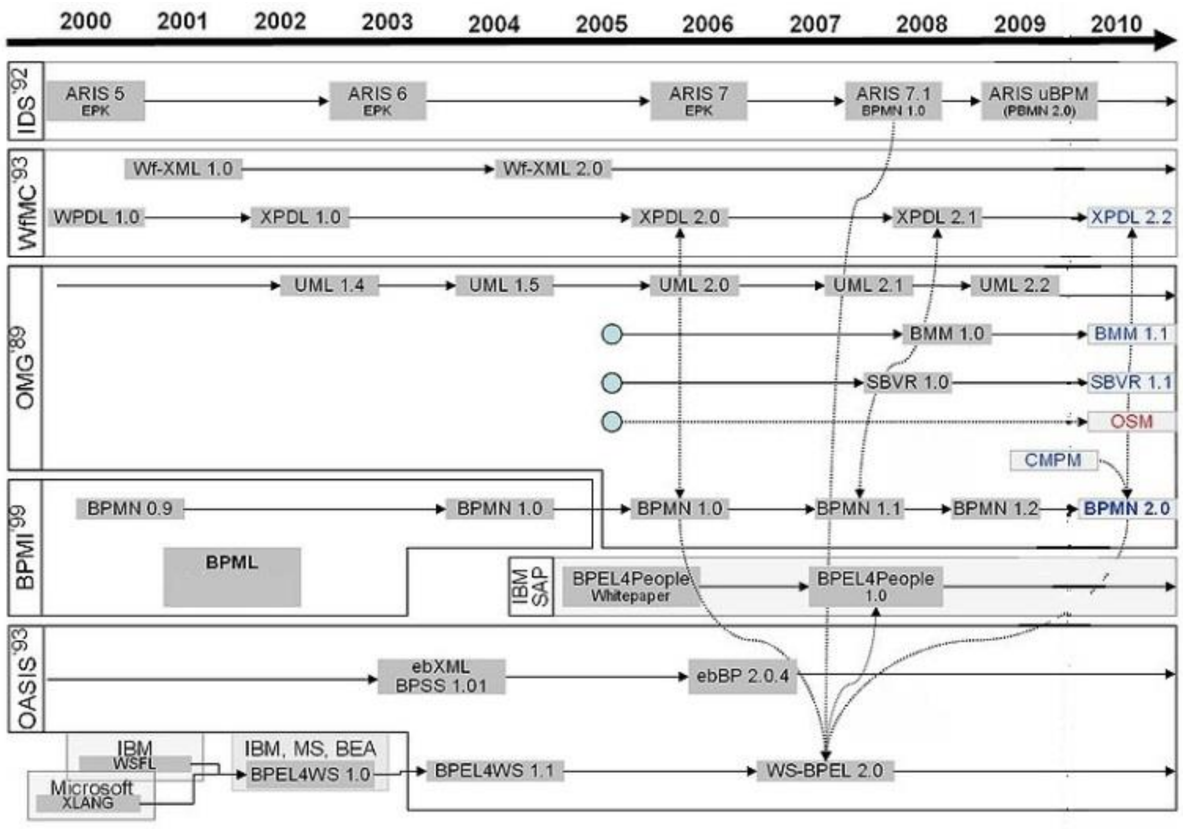


Fig. 12 - Business Process Tools [Bar2010 p. 18]

Many standards for the management and description of business protocols were defined and dismissed over the last decade (cf. Fig. 12). Business Process Management Notation (BPMN) [Mar2003] is a graphical user interface of a BPM system of OMG. Another graphical support of the Business Process Management offers UML (especially Activity Diagrams). Other BPM tools support the B2B relationship description (e.g. RosettaNet [TrPrCo2003], or the Business Process Specification Schema (BPSS), which is part of ebXML standards). Business Transaction Protocol (BTP) [OAS2010] is a protocol for managing complex, B2B transactions over the Internet.

For the "execution" of business processes, they have to be related with existing services. The XML Process Definition Language (XPDL) of the Workflow Management Coalition (WfMC) and the Business Process Modelling Language (BPML) [OAS2003] of the Business Process Management Initiative (BPMI) are languages for the process "execution" of BPM models. The ebXML standards also defined their process execution environment (cf. section 3.5). However, BPML is described as obsolete in [Ko2009] and the globally accepted standards for process "execution" are now Web Service standards like WS-BPEL of OASIS as an orchestration language and WS-CDL as a choreography language (cf. Fig. 12). These standards are discussed in the next section.

3.5 Web Service Composition (WSC)

Web Service Composition can be done on service level with the WSDL descriptions (section 3.3) or their semantic enhancements, the Capability Descriptions (section 4.3). For the Web Service Composition on process level further standards were introduced, in order to integrate the execution of business processes designed with BPMs middleware (cf. section 3.4). Examples of these Web Services description standards are Process Specification Language (PSL) of the National Institute for Standards and Technology (NIST) [ISO2004] and Business Process Execution Language (BPEL4WS, WS-BPEL, BPEL) [AndCur et al.2003; IBM2007b; ManMcI2003; Vas2007]. Both standards define orchestration languages, i.e. they describe the process of a service with its invocation of other services from the local perspective of the service. PSL was developed for addressing the semantic challenge (section 4) of orchestration languages. It is based on RDF and OWL. The terms of its described business processes are semantically enhanced [GrüMen2003].

BPEL became a popular standard orchestration language for BPM workflow orchestrations (cf. Fig. 12). BPEL4WS (Business Process Execution Language for Web Services) (BPEL for short) [AndCur et al.2003; IBM2007b] is a process modelling language supporting abstract and executable processes. The introduction of abstract processes is useful for describing business processes (cf. Fig. 9) in general; while executable processes may be compiled into invokable services. Composite Services are modelled as directed graphs where the nodes are services and the edges represent a dependency link from one service to another. Canonical programmatic constructs (like e.g. SWITCH, WHILE, and PICK) allow directing an execution's path through the graph.

BPEL was released along with two other specifications: WS-Coordination and WS-Transaction, in order to achieve transaction control and coordination of Concrete Services. These standards are discussed in deep in [AloCas et al.2004]. WS-Coordination describes how services can make use of predefined coordination contexts, in order to subscribe to a particular role in a collaborative activity. For instance distributed applications operating in a heterogeneous environment can create a context, in order to propagate an activity to other services and to register for coordination protocols (cf. [KumNan2005]). WS-Transaction provides a framework for incorporating transactional semantics into coordinated activities. WS-Transaction uses WS-Coordination to extend BPEL, in order to provide a context for transactional agreements between services. BPEL4D of Bohn [Boh2009] extends BEPL for the support of processes of technical services accessing devices. BPEL4People [IBM2007a] integrated people into the workflow processing. BPTX [Xia2007] supports long running transaction.

The choreography of Web Services was supported through several standards. Early on, a conversation model of component integration using Web Services was developed. Such a model was supported through the specifications of CS-WS (Conversation Support for Web Services). The model contained so-called Conversation Policies (CP), which describe the message formats as well as timing and sequencing constraints of the involved Web Services. The standard Web Services Conversation Language (WSCL) [BanBar et al.2002] as a choreography language introduced so-called abstract interfaces of Web Services. These abstract interfaces allowed the hiding of the inherent complexity of services. For instance, some company policies demand this encapsulation.

More recent standards for the support of choreography are grouped under the name WS-choreography. WS-choreography standards [W3C2004] describe the choreography of Web Services separately. In this context, two standards were introduced: Web Service choreography Interface (WSCI) [ArkAsk et al.2002; BroCan et al.2004], and Web Services choreography Description Language (WS-CDL) [KaBuRi2004; KavBur et al.2005; KaWaHu2007]. WSCI (XML interface language for interaction between Web Services) allows the description of client-server relationships with a local choreography (cf. section 3.4). WS-CDL supports peer-to-peer communication with a global choreography (cf. section 3.4). WS-CDL became popular, whereas WSCI, WSCL, and CS-WS are described as obsolete in [Ko2009].

A choreography description language like WS-CDL permits the descriptions of how Web Services can be composed, how service roles and associations in Web Services can be established, and how the state, if any, of composed services is to be managed. The research of Kang [KaWaHu2007] extends the choreography language to WS-CDL+ for choreography of applications in general. Tools allow the generation of single process orchestrations (e.g. BPEL) from a choreography language, i.e. a view on the choreography from the perspective of a specific service (local choreography).

Web Services were not the first attempt of standardization of e-services for business. An example for earlier standards for e-services over Web was ebXML (Electronic Business using XML) [OAS2006a] (Fig. 8). The ebXML standards were advanced by OASIS and UN/CEFACT and approved as ISO 15000. Similar like the standards of Web Services, their goal was to enable enterprises of any size and in any geographical location to conduct business over the Internet. The holistic and top-down defined approach of ebXML is an all-in-one solution and places emphasis on the business and its processes. The definition of the ebXML standards started before the establishment of the Web Service standards. Nevertheless, some Web Service standards like e.g. SOAP were integrated. The standards of ebXML are ready for deployment and during the first years of experience the specifications have matured. In contrast, Web Service specifications have been usually developed independent of each other, leading to a sometimes incoherent but flexible technology. Additionally, Web Services have a good vendor support since all major players on the IT market engage in SOA. A comparison between the standards of Web Services and ebXML can be found in [Ger2006].

The standards of ebXML directly support orchestration and choreography (EAI aspects) and security features. They define Collaboration Protocol Profiles (CPP) as well as Collaboration Protocol Agreements (CPA). The ebXML services are advertised through CPP descriptions in a common repository; the CPA functioned as Service Licence Agreement (SLA). In this way, the Service Discovery of ebXML is more precise than the one of Web Services through WSDL descriptions in UDDI repositories. However, the standards of ebXML do not tackle the semantic challenge. In some areas, interoperability brings the two approaches together, for example, by using the Universal Business Language to describe business documents or by following the UN/CEFACT Modelling Methodology to acquire knowledge about business processes. After all, the main difference will remain that the top-down design approach of ebXML will continue to yield different results and addresses different audience than the bottom-up approach of Web Services.

3.6 Cloud Computing- offering computing resources as service

Since 2007, the term Cloud Computing has moved mainstream in SOC (cf. [GraMax et al.2010]) due to a large scale research project of Google, IBM, and a number of other universities. It propagates an approach of providing a transparent access of computing resources and virtual IT-infrastructure as service in an abstract, dynamically needs adapted and accountable way. The accounting shall be on-demand. The transparent character of the usage, provided in a not further specified part of a network (mostly Internet or intranet), is reflected in the metaphor or symbol “cloud”. Cloud Computing is explained as an approach, which goes beyond Software-as-a-Service (SaaS), Organic Computing [Org2010] and Virtualisation (methods for accessing computing resources transparently).

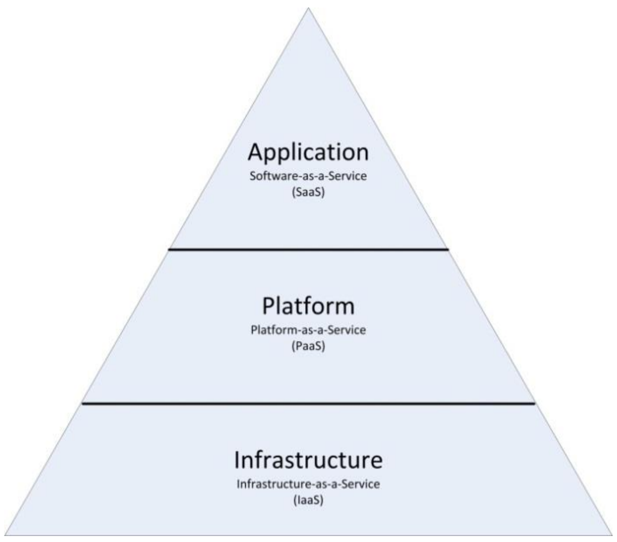


Fig. 13 - Service Models of Cloud Computing

Cloud Computing is still an evolving paradigm and no commonly accepted definition exists until now. A popular definition of the National Institute of Standards and Technology (NIST, [NIS2010]) states: *Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or Service Provider interaction.*

NIST defines three Service Models of Cloud Computing (Fig. 13): (1) Infrastructure-as-a-Service (IaaS), (2) Platform-as-a-Service (PaaS), and (3) Software-as-a-Service (SaaS). The infrastructure layer, also called "Cloud Foundation", is the lowest Service Model layer of Cloud Computing. Together with the next layer PaaS, IaaS is also called "Cloud Housing". IaaS provides an environment of virtual servers. An example is Elastic Compute Cloud (EC2) [ama2009] of Amazon. The advantage of using this infrastructure as a service in comparison to a traditional data centre is the scalability. In principle, the Service Client can increase and reduce the number of virtual servers on demand. He has full access rights on the virtual hardware and can install new applications. However, the customer must also do the system administration.

The platform layer (PaaS) supplies a server for an application of the Service Client. The customer copies the application onto the server, in order to get it executed. The application acquires the physical resources transparently. The Service Client does not have to do any system administration any longer. Examples are Windows Azure of Microsoft, App Engine of Google, and force.com of Salesforce.com.

The application layer represents the top model of the Service Models (SaaS) of Cloud Computing. The customer can use an application provided in the cloud, which is modular, service-oriented, dynamic, distributed, and multi-client enabled. Since the success of the application can possibly not be quantified by the Service Provider, the accounting according with the load is difficult. Known examples of SaaS in Cloud Computing are Google Docs as well as Microsoft SkyDrive Office Web Apps, Exchange Online, SharePoint Online, Office Communication Online, and LiveMeeting.

One can distinguish between three main kinds of cloud: (1) Private Cloud, (2) Public Cloud, and (3) Hybrid Cloud. In a Private Cloud, services are provided and consumed by actors of one company (Enterprise Cloud). In this way, the effort for the system management is kept low and the cloud could be used for experiments (Exploratory Cloud). For security reasons, the number of actors could be further reduced to the members of one department (Departmental Cloud). A Public Cloud allows principally everybody to be a Service Client or Service Provider. However, data security and system administration become more difficult. A Hybrid Cloud is likely a Private Cloud, which becomes temporary a Public Cloud, when additional resources have to be allocated in periods of high resource demand.

In the case a customer is an employee of a small company and the cloud service is not used permanently, then the advantages of the customer should be firstly a financial gain through on-demand accounting, secondly, the saving of additional local resources, and thirdly an efficient use of resources. Critics argue that the data security would be endangered with curious Service Providers like Google. National and international laws have to be considered and updated. Additionally, performance problems can occur, when the customer wants to have an effective data encryption. Finally, a customer will likely become locked with one Service Provider since the cloud interfaces are not standardised (Lock-in-Effect).

Cloud Computing can be delimited from similar terms. Grid Computing intends the common using of computing resources, which have no central control. Cloud Computing possesses a Service Provider that centrally controls, provides, and offers resources, which are accessed by one Service Client. Similar, peer-to-peer networks have the purpose of taking load of the servers instead of offering the resources as a service like Cloud Computing does. Critically looked at the definition of Cloud Computing, it turns out that IaaS as well as PaaS have their application logic for transparent providing and accounting of the infrastructure or the platform, respectively. From this point of view, they are both SaaS with goal-restricted applications. In the end, the idea of SOC goes beyond Cloud Computing, since it involves application integration, i.e. the composition of services.

3.7 Summary

SOA solutions are the next evolutionary step in software architectures. SOA is an IT architecture, in which functions are defined as independent services with well-defined, invokable interfaces. An important issue in today's design of software architectures is to satisfy increasing software complexity and flexible solutions. Classic EAI projects are expensive and inflexible due to costs, proprietary solutions, and tightly coupled interfaces. The Service Discovery and an improvement of the Service Design become increasingly important.

However, the centralised Service Discovery with UDDI, no semantic enhancement of WSDL descriptions, the weak security of SOAP, the high number of standards, and the incompatibility between different versions of the standards were recognized as weaknesses of Web Services. Especially, the support of BPM leads to the introduction and dismissing of several standards. The recognized weaknesses of Web Services are concern of further research.

The discussed standards WS-Security, WS-Trust, and WS-Addressing improved the security weakness. However, these standards produce a number of factors that a client has to consider before interacting with a service. In order to overcome the resulting integration challenge, an extension of WSDL was introduced with the standard WS-Policy [VedOrc et al.2007]. It defines a set of generic constructs for defining and grouping policy assertions, which represent alternative sets of possibly optional requirements of the service for the interaction. For autonomic computing, the client must be able to interpret these policy assertions and adapt its behaviour accordingly.

The high number of Web Service standards and their incompatibility of versions was addressed through WS-Interoperability (WS-I) [OAS2011], which introduced so-called profiles containing sets of compatible standards and their versions. In order to ensure interoperability of Web Services, they can declare their conformance with a certain profile version. The versions of the profiles are often declared in a way that services of a lower version are still interoperable.

Keyword-based Service Discovery with the centralised repository of UDDI became unpopular, because of an unreliable identification of potential services and a costly, mostly non-existing management of unavailable services (cf. [FeKeZa2008 section(s) 8.3]). The first UDDI Business Registry (UBR) nodes, run by IBM, Microsoft, SAP, and NTT Com, were shut down at the beginning of 2006. Alternatively, Web Services can be described on various levels of abstraction for the Service Discovery. The publication of [FeKeZa2008] delimits discovery based on keywords, simple semantic descriptions, and rich semantic descriptions. Distributed repositories, peer-to-peer networks, and MAS as alternative approaches are discussed (cf. [RamHol et al.2009]). WS-Discovery [ModKem2009] describes a distributed repository with multicast addressing of service groups. The research in SOC was extended to advanced aspects of services: semantically enhanced functional descriptions (e.g. IOPE capabilities), non-functional aspects (hybrid Service Discovery), context-awareness, and Autonomic SOC. The resulting semantic challenge and the automation of SOC are discussed in the next two chapters.

4 The semantic challenge

The semantic challenge is to avoid semantic mismatches and to overcome the heterogeneity of data, in order to have a more reliable Service Discovery (cf. [Syc2010; VetLen2005]). The Semantic Web Services (SWS) research is concerned with the semantic challenge. It is based on the research of Semantic Web, which introduces ontologies for the semantic categorization of terms.

It was recognized that the Service Composition on process level of BPEL based on WSDL leads to a complex Service Discovery process. Therefore, the functionality offered by a given service is described in SWS not any longer with its messages, operations, or orchestration, but through its **Capability Description**, which contains semantically enhanced elements. The semantic categorization of the elements is done through the application of ontologies. The Service Capability is meant primary for discovery and selection purposes, i.e. the capability is used on the one hand by the Service Provider for the advertisement of the service functionality. On the other hand, the Service Requester takes advantage of the Service Capability, in order to determine whether the service meets its functional needs. Additionally, the Service Capability allows a Service Composition on service-level. The next sections look at ontologies and Service Capabilities. Approaches of SWS are discussed in section 4.4.

4.1 Ontology

Ontologies are commonly used in artificial intelligence and knowledge representation for conceptual classification schemes. The computer science usage of the term ontology is derived from the much older usage of the term in philosophy, where it means the study of being or existence. In the context of computer and information sciences, an **ontology** is an explicit formal specification of a shared conceptualisation. It defines a set of representational primitives with which to model a domain of knowledge or discourse (cf. [Gru2009]). The ontologies are organized by concepts (also called classes or sets), properties (or attributes), and relationships. An ontology contains besides a hierarchy of concepts organized by the subsumption relation (often called *isa*, *subtype*, or *subclass*), additional 'semantic relations' that specify how one concept is related to another. These additional semantic relations define part-of relations and other constraints.

A categorisation of ontologies themselves can be made according to their subject of conceptualisation (cf. [StGrAb2007]): Top-level Ontology, Domain Ontology, Task Ontology, and Application Ontology. A **Top-level Ontology** (also called upper ontology or foundational ontology) attempts to describe very abstract and general concepts that can be shared across many domains and applications. Due to their generality, they are typically not directly used in applications but for other ontologies to be aligned to. **Domain Ontologies** capture the knowledge within a specific domain of discourse, such as medicine or geography. **Task Ontologies** depict the knowledge about a particular task, such as diagnosing or configuring. Further narrowing the scope, Application Ontologies provide the specific vocabulary required to describe a certain task enactment in a particular application context. **Application Ontologies** make use of both Domain and Task Ontologies. They describe e.g. the role that some domain entities play in a specific task.

Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic level", whereas database schema are models of data at the "logical" or "physical" level [Gru2009]. By using ontologies to enrich the description of services, their semantics become machine-interpretable, and users are enabled to pose concise and expressive queries. Furthermore, logical reasoning can be used to discover implicit relationships between search terms and Service Descriptions as well as to flexibly construct taxonomies for classifying services. **Mediation** allows the translation between related ontologies and the adaptation of terms of distinct domains or user perspectives.

On top of RDF and RDFS, languages based on XML, W3C standardisation efforts have produced the OWL family of Ontology Languages for the description of Semantic Web ontologies. Ontology Languages are based on logics like Description Logic (DL [FraDie et al.2005]) (e.g. OWL-DL [BecHar et al.2004]), Frame-Logic (F-Logic, ObjectLogic [KiLaWu1995]) (e.g. WSMO/WSML [BruBus et al.2005], SWSO/SWSL [BatBer et al.2005]), First Order Logic (FOL) and others. In contrast to description logic based formalism, the semantics of F-logic are normally that of a closed world assumption. F-logic is generally undecidable; whereas DL based on *SHOIN* logic is decidable and with an open world assumption. F-Logic is more expressive than DL.

First-Order Logic (**FOL**) is used for annotating and matching pre- and postconditions of operations. FOL is a branch of logic that is based on individuals and the relations (predicates) between them. FOL permits the formulation of quantified statements about some or all the individuals in the universe of discourse. Predicates in FOL take only individuals as arguments and quantifiers only bind individual variables. The goal of logic inference in FOL is to check whether a given knowledge base KB (a collection of sentences) entails a sentence A ($KB \vdash A$), i.e., whether A follows logically from KB. Entailment in FOL is semidecidable, i.e. every entailed sentence can be found, but for non-entailed sentences, it is not always possible to decide whether they are entailed or not, because the logic inference might not finish. Despite these theoretical limits, automated theorem provers can solve many hard problems in FOL. Inference procedures often employed include resolution and term rewriting.

Description Logics (**DL**) are a family of knowledge representation languages, which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. The name description logic refers, on the one hand, to concept descriptions used to describe a domain and, on the other hand to the logic-based semantics which can be given by a translation into first-order predicate logic. DL was designed as an extension to frames and semantic networks, which were not equipped with formal logic-based semantics. Description Logic was given its current name in the 1980s. Before this, it was called (chronologically): terminological systems and concept languages. The first DL-based system was KL-ONE (by Brachman and Schmolze, 1985).

DL	FOL
$E \equiv D$	$\forall x: (E(x) \leftrightarrow (D(x))$
$E \sqsubseteq C \sqcap \neg D$	$\forall x: (E(x) \rightarrow (C(x) \land \neg D(x)))$
$\forall R.C$	$\forall y: (R(x,y) \rightarrow C(y))$
$\exists R.C$	$\exists y: (R(x,y) \land C(y))$

Table 4 - Comparison between DL and FOL

Generally, Description Logic is a subset of First Order Logic (cf. Table 4). It consists of an ABox and a TBox. The TBox contains rules based on atomic concepts (predicates of arity one) and atomic roles (predicates of arity two). ABox contains objects, i.e. instances of the concepts. The semantic level is defined through an interpretation $I = (\Delta^I, \circ^I)$. The set Δ^I is the domain set of all objects in the ABox. The symbol $\circ^I$ is a mapping of objects of Δ^I to the atomic concepts and atomic roles, respectively. An atomic concept A is mapped with a subset $A^I \subseteq \Delta^I$. The mapping of an atomic role P is a subset $P^I \subseteq \Delta^I \times \Delta^I$. The TBox contains concepts (e.g. C, D, E) as well as roles (e.g. R), which defined through concept definition $C \equiv D$, concept inclusion $C \sqsubseteq D$, and concept inverse inclusion $C \sqsupseteq D$ with the constructors and semantics given in Table 5. The inclusions define hierarchies of concepts.

DL comprises different kinds of description logics. One important and influential description logic is called *ALC*. *ALC* introduces the special concepts *nothing* $\bot$ and *thing* $\top$, ($\bot \equiv C \sqcap \neg C$, $\top \equiv \Delta \equiv C \sqcup \neg C$). It supports the operations *Negation of concepts*, *Conjunction*, *Disjunction*,

Syntax	Description	Semantic
$E \equiv \neg C$	Negation of concepts	$(\neg C)^I = \Delta^I \setminus C^I$
$E \equiv C \sqcap D$	Intersection, Conjunction	$(C \sqcap D)^I = C^I \cap D^I$
$E \equiv C \sqcup D$	Disjunction	$(C \sqcup D)^I = C^I \cup D^I$
$E \equiv \forall R.C$		$(\forall R.C)^I = \left\{\begin{matrix} e \in \Delta^I \mid \forall e' : (e,e') \in R^I \rightarrow \\ e' \in C^I \end{matrix}\right\}$
$E \equiv \exists R.C$	Value restriction (existential)	$(\exists R.C)^I = \left\{\begin{matrix} e \in \Delta^I \mid \exists e' : (e,e') \in R^I \wedge \\ e' \in C^I \end{matrix}\right\}$
$E \equiv \exists^{\odot n} R.C$	Number restriction	$(\exists^{\odot n} R.C)^I = \left\{\begin{matrix} e \in \Delta^I \mid \odot \in \{=,\leq,\geq\}, n \in \mathbb{N}, \\ \#\{e' \mid (e,e') \in R^I \wedge e' \in C^I\} \odot n \end{matrix}\right\}$
$E \equiv \exists^{\odot n} R$	General Number restriction	$(\exists^{\odot n} R)^I = \left\{\begin{matrix} e \in \Delta^I \mid \odot \in \{=,\leq,\geq\}, n \in \mathbb{N}, \\ \#\{e' \mid (e,e') \in R^I \wedge e' \in \Delta^I\} \odot n \end{matrix}\right\}$

Table 5 - Constructors of DL

and *Value restriction* (Table 5). Domain and Range of a role R can be expressed with the value restrictions (Domain: $\exists R.\top \sqsubseteq \mathrm{domain}$, Range: $\top \sqsubseteq \forall R.\mathrm{range}$).

An extension of *ALC* is the description logic *SHOIN(D)*:

- Roles have additional properties like being transitive, symmetric or inverse to other roles.
- Roles can be arranged hierarchically.
  The formula $R \sqsubseteq S$ in DL is translated in FOL $\forall x,y: (R(x,y) \rightarrow S(x,y))$.
- Individuals in the ABox can be compared.
- Nominals, i.e. concepts, which directly enumerate (or restrict) their individuals, are possible in the TBox.
- (General) Number restrictions are possible in the TBox (Table 5).
- Besides **abstract roles**, additionally concrete roles are allowed, which can have an assignment of datatype values such as integers of strings to individuals.

4.2 Capability Descriptions with DL

In their Capability Description, requesters and providers of services want to express which service instances they are willing to accept, i.e. which ones they request or provide, respectively. In order not to list all the different services explicitly, they take use of Abstract Service Descriptions. Describing a set of objects in DL is done by using concepts. In this way, the set of Concrete Services described by an Abstract Service is the set of service instances acceptable to a Service Requester or Service Provider, respectively. With an interpretation I, these service instances map to the extension S^I of a DL concept S that represents the Abstract Service. The concept S is specified by a set of DL axioms D, which can be associated with the Capability Description of a service.

In equation (4-1) the possible axioms D_P of a "shipping" Service Provider are listed. They specify the concept S_P: "Shipping items from any city in UK to a city in Germany, which have a weight less than 50 kg". The second axiom of D_P assures through general restrictions that the concept $Shipping$ really has the properties of going from exactly one city to another one. The Concrete Services S_P^I could contain among others the two service instances for shipping a 50 kg package from Plymouth to Bremen and for shipping a 25 kg barrel from Dover to Hamburg.

$$D_P = \left\{ \begin{matrix} S_P \equiv Shipping \sqcap \forall item.(\forall weight.\leq_{50}) \sqcap \exists from.UKCity \sqcap \\ \exists to.GermanCity, \\ Shipping \sqsubseteq (\exists^{=1} from \sqcap \exists^{=1} to), \qquad UKCity \sqsubseteq \exists from^{-1}.S_P \end{matrix} \right\} \tag{4-1}$$

It becomes obvious that the roles declare properties of services and their constraints. The **value variety of properties** can be fixed to certain values with nominals (e.g. $\forall R.\{i\}$, $\forall R.\leq_n$ like $\forall weight.\leq_{50}$ in D_P) or ranging over the instances of a certain concept (e.g. $\forall R.C$). Obligatory properties are introduced with an existence restriction (e.g. $\exists R.C, \exists R.\top$ like $\exists from.UKCity$ in D_P). An axiom with $(\exists R.\top)$ does not restrict the value of an obligatory property R. Number restrictions of the roles cover the multiplicity constraints of properties (e.g. $Shipping \sqsubseteq (\exists^{=1} from \sqcap \exists^{=1} to)$ in D_P). Through the number restriction ≤ 1 a property can be declared as a single-valued property. Alternatively, an axiom can declare a multi-valued property. The axioms in D can also contain for the covered services a range-covering of a property through an axiom like $C \sqsubseteq \exists R^{-1}.S$. Such an axiom with the use of an inverse role means that in every possible world (i.e. in all interpretations I) for any instance or value x belonging to the concept C^I a service instance y in S^I exists, which holds as property R this instance. Translated in FOL, this axiom can be written as $\forall x: (C(x) \rightarrow \exists y: [R(y,x) \wedge S(y)])$. In D_P the axiom $UKCity \sqsubseteq \exists from^{-1}.S_P$ declares that the services are offered from any city in the UK.

$$KB = \{Shipping \sqsubseteq \exists^{=1} from, \qquad UKCity(Plymouth, London)\} \tag{4-2}$$

$$D_R = \left\{ \begin{matrix} S_R \equiv Shipping \sqcap \forall from.\{Plymouth, Dublin\}, \\ \{Plymouth\} \sqsubseteq \exists from^{-1}.S_R, \\ \{Dublin\} \sqsubseteq \exists from^{-1}.S_R \end{matrix} \right\} \tag{4-3}$$

The axioms D_P can be used by the Service Provider, in order to advertise his shipping services. In a similar way, the Service Requester could define axioms D_R for the Capability Description of the requested services. Both sets of axioms can be used by a DL reasoner, in order to determine matching of the capabilities by inference. The matching algorithm is based on a common knowledge base KB and the two Capability Descriptions D_R and D_P. The Service Requester in (4-3) wants to find a Service Provider, who offers services, which go from Plymouth and Dublin. The KB in (4-2) declares Plymouth and London as belonging to the concept UKCity.

In the chapter "Discovery" [StGrAb2007 pp. 211–244], Grimm discusses three alternatives of matching for a DL reasoner using the sketched model (Table 6):

(1) intersection matching ($match_{int}(KB, D_R, D_P)$),

(2) subsumption matching in both ways ($match_{sub\Rightarrow}(KB, D_R, D_P)$, $match_{sub\Leftarrow}(KB, D_R, D_P)$),

(3) and non-disjointness matching ($match_{ndj}(KB, D_R, D_P)$).

However, matching based on logical inferencing is computationally costly and demands high-quality semantic Service Descriptions. To realise a practical discovery framework for large-scale real-world scenarios, the different techniques for matching and retrieval needs to be combined appropriately with regard to architectural issues.

Function	**Formula**	**Intuition**
$match_{int}(KB, D_R, D_P)$	$KB \cup D_R \cup D_P \models (S_R \sqcap S_P)$	Is there a way to resolve unspecified issues such that D_R and D_P specify some common service instances?
$match_{sub\Rightarrow}(KB, D_R, D_P)$	$KB \cup D_R \cup D_P \models (S_R \sqsubseteq S_P)$	Do the service instances of D_P encompass the service instances of D_R regardless of how unspecified issues are resolved?
$match_{sub\Leftarrow}(KB, D_R, D_P)$	$KB \cup D_R \cup D_P \models (S_P \sqsubseteq S_R)$	Do the service instances of D_R encompass the service instances of D_P regardless of how unspecified issues are resolved?
$match_{ndj}(KB, D_R, D_P)$	$KB \cup D_R \cup D_P \not\models (S_R \sqcap S_P \sqsubseteq \perp)$	Do D_R and D_P specify some common service instances, regardless of how unspecified issues are resolved?

Table 6 - Alternatives of matching for DL based Capability Descriptions

For the equations (4-1) to (4-3), the matching $match_{int}(KB, D_R, D_P)$ holds, because the concept $S_R \sqcap S_P$ is satisfiable with respect to $KB \cup D_R \cup D_P$ since Plymouth is a UK city, and it is in the range of the $from$-role of both (D_R and D_P). The matching $match_{ndj}(KB, D_R, D_P)$ also holds, because in all interpretations the given concept is satisfiable, due to the definition of Plymouth as an $UKCity$ in KB. In [StGrAb2007], the Non-Disjointness matching is described with some advantages, but it has to use range-covering and it hits some deficits of expressiveness of DL. The subsumption matching alternatives do not hold for the equations. Grimm illustrates in [StGrAb2007] the advantages of all matching alternatives and their possible combinations.

4.3 Capability description in SWS

For Capability Descriptions of services, various SWS annotation frameworks include information about input and output parameters, state-transition-based notions, explicit taxonomic classification, or high-level abstract properties of a service. The Capability Descriptions allow a service-based Service Composition like shown in [FujSud2009] or [Hab2009]. Ontologies are used for the realisation of semantically enhanced categorization of parameter types of the service operations and messages as well as for the semantic enhancement of constraint properties. State-transition-based notions, used for the behavioural description of the operations, go back on Hoare logic. The Capability Descriptions of SWS approaches, namely SAWSDL, OWL-S, and WSMO, are subsequently discussed.

Hoare logic is a formal system that provides a set of logical rules for reasoning about some properties of a computer program, including determining whether a given program provides a formally defined functionality. The intended functionality of a program Q is specified in terms of initial preconditions (P), i.e., assertions about certain properties of the values taken by the relevant variables before the program initiation and the relations among them, and postconditions (R), i.e., assertions about the values after execution. The relation between the preconditions and postconditions is formulated as so-called Hoare triples of the form P[Q]R which can be interpreted as follows: "If the assertion P is true before initiation of a program Q, then the assertion R will be true on its completion." Specifications based on pre- and postconditions can be used for discovering software components or services with a required functionality.

In OWL-S, the Capability Description can be described in short as **IOPE** (Input, Output, Precondition, and Effect). IOPE was already known to Web Services independent trading approaches like LARKS [SycWid et al.2002]. The set Input contains the type declarations of the input parameters which are necessary for the execution of the services, whereas the set Output contains the declarations of variables, which are an output of the application of the service. SAWSDL just extends the functional descriptions of WSDL with XML tags for the preconditions and effects. However it is missing the inclusion of an ontology language like for instance OWL in the standards of OWL-S. The Precondition and Effect can be compared with preconditions and postconditions of the Hoare logic.

The capability class in WSMO consists of the four elements *hasPrecondition*, *hasAssumtion*, *hasPostcondition*, and *hasEffect*. The *hasPrecondition* and *hasPostcondition* expressions make axiomatic statements about the expected input and output variables, i.e. WSMO does not only enumerate the declarations, but states with logic expressions what information must be available for the service to be executed and what information will be available after the service has been executed. The *hasAssumption* and *hasEffect* expressions are again comparable with the preconditions and postconditions of the Hoare logic. They make statements about the assumed state of the world prior to the execution and the guaranteed state of the world afterwards.

Generally, the standards of SWS discovery frameworks define the domain-independent part of the ontology vocabulary in terms of which Capability Descriptions are to be defined. In order to completely describe the capability of a service, additional vocabulary originating from a Domain Ontology becomes necessary. For instance a service within a logistic domain would include additional concepts like "Transportation", "Container", or "Location" for a less ambiguous Capability Description.

The decision of using WSDL as Service Description or the semantically enhanced Service Capability belongs to the Service Design of a service. Every Discovery Agent has to implement an appropriate search and matching algorithm. The Service Design and the Service Discovery are phases of the life cycle of a service, which is introduced in the next section. The algorithms of the phases and the inclusion of specific Domain Ontologies mean additional effort and ambiguity in the application of SWS standards.

4.4 Predominant Discovery Approaches in SWS

The motivation for the research in Semantic Web Services (SWS) was the recognition that the centralised and functional discovery of Web Services based on WSDL and UDDI as well as their composition can be improved through semantic annotation based on ontologies [Gri2007]. Hartmann [HarSur2004] shows that, for practical implementations, Semantic Web technologies must consider aspects such as scalability and reliability. There are various efforts that investigate the different techniques of Semantic Web Services in the context of Service Discovery. Many of them are tightly coupled with the Service Capabilities (cf. section 3.7). Results of this research in SWS are standards like SAWSDL (Semantic Annotations for WSDL and XML Schema) [FarLau2007b] (formerly WSDL-S [AkkFar et al.2005]), WSMO [BruBus et al.2005], and OWL-S [MarBur et al.2008], which are built on the standards of Web Services. A review of Semantic Web Service Discovery methods can be found in [LeKiKa2010].

4.4.1 OWL-S (formerly DAML-S)

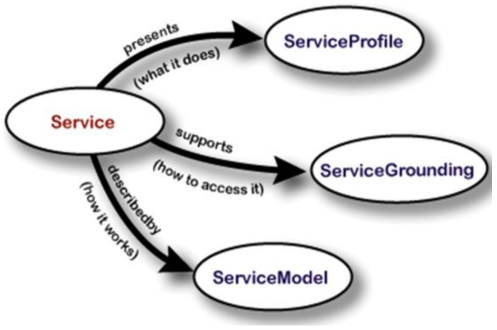


Fig. 14 - The models of OWL-S

The Defence Advanced Research Projects Agency (DARPA) developed a markup language for the description of ontologies, the DARPA Agent Markup Language (DAML), which was based on DL. In the process of standardizations by the World Wide Web Consortium (W3C), DAML was renamed to Ontology Web Language (OWL). **OWL-S** (formerly DAML-S) is an approach for providing an ontology, which allows the description of Web Services. The OWL-S Service Profile is a representation of the operations provided by the service. Service Descriptions are instances of the (static) OWL concepts of the Service Profile. Top-level ontologies can be defined for the parameters in the functional description and for the Service Profiles themselves. OWL-S based approaches use the Service Profiles and Domain Ontologies, in order to decide the matching of Service Request and Service Advertisement (cf. [PaoKaw et al.2002]). However, the standard of OWL-S does not specify specific matching algorithms or Domain Ontologies, nor does the standard define a data format specific for the Service Request. In [SinHuh et al.2005], they show that the Service Profile can be integrated into UDDI with the t-models.

OWL-S/UDDI [OAUDxm2008] matchmakers use the Service Profile for Service Discovery with UDDI. However, OWL-S offers more than just advertisement and discovery; it also integrates other aspects of the life cycle with the Service Model and Service Grounding.

Control Flow Construct	Meaning
Sequence/Unordered	The constructs define a list of processes that are executed in sequence or in a random order.
Conditionals	if-then-else statements
Loops	while and repeat-until statements
Multithreading and synchronization	These constructs split the process in multiple threads, and rendezvous (join) points
Non-deterministic choices	Constructs allow an (arbitrarily) select of a process from a set.

Table 7 - Control flow constructs of workflows in OWL-S

The Semantic Web Service standards of OWL-S [MarBur et al.2008] reflect the introduced life cycle in Fig. 2. The service concept in OWL-S links the Service Profile, Service Model, and Service Grounding (in Fig. 14 the names of the models and their relationships are shown as they appear in the markup language). A Service Profile represents a service, which is described by a Service Model. The service supports a Service Grounding. The Service Profile model contains the Service Description for the Service Discovery. The Service Composition on process level is

Aspect of Services	Comment
1) View	• no special Service Requester view • Hierarchy of Service Profiles possible – could be used in the direction of domains
2) non-functional attributes	• Only the service profile contains non-functional attributes: name, description, and actor. However, there are no special concepts for these attributes.
3) Inherent Complexity	• Workflow support • service-level composition • orchestration like BPEL
4) life cycle	Phase 1: • Service profile partly based on the Process Model for Service Description • IOPE Service Capability • Ontologies for parameter types • Ontology for service profiles Phase 2: UDDI extension for OWL-S Phase 3: No special matching standardised Phase 5: Grounding support through model Phase 6: WSDL-files linked with Grounding model

Table 8 - Classification of OWL-S through the four aspects

described by a Process Model, which is an instance of the Service Model. The OWL-S Service Model is the description of the service. Atomic Processes in the Process Model of OWL-S are linked through the Service Model with a service, which has a support link to the real service most often accessed through its WSDL file. In the Process Model processes can be chained to form a workflow. OWL-S includes the control flow constructs shown in Table 7.

4.4.2 Semantic Web Services Framework (SWSF)

The Semantic Web Services Framework (**SWSF** [BatBer et al.2005]) consists of the ontology SWSO and the language SWSL. SWSO can be seen as an extension and refinement of OWL-S due to the underlying richer language SWSL (in comparison to the language OWL DL), which describes more in detail the Service Composition on process level with the Process Specification

Aspect of Services	Comment
1) View	Even more business oriented than OWL-S
2) non-functional attributes	
3) Inherent Complexity	Business process support
4) life cycle	Phase 1: Own language Phase 3: Open for extended querying

Table 9 - Classification of SWSF (extension of OWL-S)

Language (PSL [ISO2004]). The Service Model of OWL-S is rather an extension of BPEL and WSCI [ArkAsk et al.2002]. The publication of [BatBer et al.2005] describes a discovery use case, where the Service Descriptions are expressed with SWSL-rules, and the discovery is realised by executing rule-based queries. The querying is performed with transaction logic, which is a rule-based formalism that supports the explicit representation of change.

4.4.3 Web Service Modelling Ontology (WSMO)

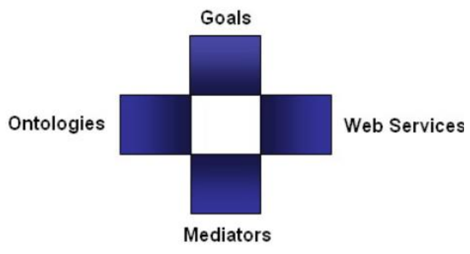


Fig. 15 - Elements of WSMO

Web Service Modelling Ontology (WSMO [BruBus et al.2005]) provides like the Service Profile of OWL-S the semantically enhanced description of a service for the Autonomic Service Discovery (in our categorisation, this enhancement of the Service Description is covered with the Service Design of the 4th aspect of services). WSMO has its conceptual basis in the Web Service Modelling Framework (WSMF [FenBus2002]). It refines and extends this framework. Additionally, it develops a formal ontology and a set of languages (WSML based on different logics). WSMO identifies four top-level elements as the main concepts of the Service Description: ontologies, services, goals, and mediators.

The ontologies provide the terminology. Ontologies are useful for the definition of domain terminology and the description of the relevant aspects of the goal and service elements. The descriptions of a service comprise the capabilities (functional view, 1st aspect), non-functional attributes (2nd aspect), and its internal working (described by a so-called interface, 3rd aspect). The Service Provision and Service Advertisement are done through the service element. The Service Request uses the goal element. WSMO distinguishes between the Service Requester and the Service Provider view (1st aspect of service). The mediators describe elements that handle interoperability problems between different WSMO elements on data, process, and protocol level (choreography).

The comprising Service Description of WSMO provides a unifying view of a service. The functional value of the service is captured by its capability (Capability Description of WSMO was compared with an IOPE capability in section 4.3). The interface description in the service element is meant for the Service Grounding (4th aspect of our categorization). It describes how the functionality of the service can be achieved by providing a twofold view: (1) choreography (decomposes a capability in terms of interaction with the service), and (2) orchestration (decomposes a capability in terms of functionality required from other services). This interface description delivers the means to interact with the Service Provider in order to request the actual performance of the service, or to negotiate some aspects of its provision.

The Service Request also contains with the goal element an interface description. It allows the Service Requester to request a certain interface in the service element, i.e. to say something about the wished choreography or orchestration. In this way, the Service Requester can influence the

Aspect of Services	Comment
1) View	• Support of Service Provider view (service) • Support of Service Requester view (goal) • Mediation between different views (mediation)
2) non-functional attributes	Every element (service, goal, mediator, ontology) has non-functional attributes, but no special data types or methods are standardised.
3) Inherent Complexity	• Interface description allows constraints for choreography and orchestration • Composition on service-level with capabilities
4) life cycle	Phase 1: • Goal and Service for a separate description of request and service • Service capability • Ontologies with mediation for constraints Phase 2: Adaption of Service Trading through links on mediation algorithms Phase 5: Constraints for negotiation through interface description

Table 10 - Classification of WSMO through the four aspects

inherent complexity of a service (this aspect of the Service Discovery environment, we covered in our categorization with the Service Grounding phase in the 4th aspect of services).

A classification of the SWS approaches OWL-S, SWSF, and WSMO through the four aspects of services is shown in Table 8, Table 9, and Table 10, respectively. The aspects of services were discussed in section 2.3. Only criteria, which make the approaches special with regard to the aspects, are listed.

4.4.4 SAWSDL (formerly WSDL-S)

Semantic Annotations for WSDL and XML Schema (SAWSDL) [FarLau2007a] is a direct extension of descriptions in WSDL with annotation tags. SAWSDL replaced WSDL-S [AkkFar et al.2005] since it includes additional annotations for the support of federated registries. As SAWSDL does not specify a language for representing the semantic models, it does not claim to be a fully-fledged description framework/ontology, i.e. it stays ambiguous in the formulation of a semantically enhanced Service Description or Service Request. However, SAWSDL provides annotation mechanisms by which concepts from the semantic models can be referenced from within WSDL, WS-BPEL, and XML Schemata components. The annotation tags contain references to semantic models of ontologies or capabilities, similar to the ones in OWL-S or WSMO. They allow Web Service developers to annotate their Web Services with their choice of ontology language (e.g. UML and OWL). SAWSDL is part of the METEOR-S project ([VerGom et al.2005], cf. 5.2.1), which compasses the direct inclusion of semantic enhancements in the standards of Web Services for the support of autonomic service computing. SESMA [Pee2005] was an alternative approach in this direction.

4.4.5 Selected Approaches of SWS

In the domain of distributed open information retrieval, queries may span across multiple data resources with multiple levels of data heterogeneity and the involvement of users with multiple levels of understanding. A semantic mediation approach based on OWL is described in [Hua2008], which eases data interoperability and the tolerance of data heterogeneities. As part of this project, a decentralised directory service with QoS criteria has been developed to improve the availability of metadata repository, so that if the central directory is unavailable, distributed repositories can take over Service Discovery. The implementation is based on the MAS middleware JADE. In the same domain, the approach of [AnSaRa2008] includes Service Composition. Here the proposed algorithm makes use of OWL-S ontologies, and explicitly returns the sequence of atomic process invocations that the client must perform in order to achieve the desired result. When no full match is possible, the algorithm features a flexible matching by returning partial matches and by suggesting additional inputs that would produce a full match.

In the domain of processing of geographical data (geoprocessing), the approach described in [LutMic2007] looks for the Service Composition of data providing and geoprocessing services in a globally distributed, special data infrastructures. The approach overcomes the ambiguities of natural languages and low precision in keyword-based alternatives. The Capability Descriptions of services in OWL-S comprise geospatial operations and their requirements. The matching is based on function subtyping.

The approach of [GurZei2005; YeChe2006] involves Service Discovery, Service Composition on service-level, self-healing, and automated Service Grounding based on the deductive program synthesis theory. The implementation of a Composite Service is extracted from the proof. The IOPE Service Capabilities of services are translated into first-order logic axioms. The Service Capability of the query also transforms into a FOL formula. An automatic theorem proofer is used for the generation of the poof.

In the approach of [KvaRon et al.2005] in the biomedical domain, Grid Services are annotated with shared Domain Ontologies, algorithms for automated Service Composition with matching of Service Capabilities are proposed, and a selection of the gained workflows is based on a trade-off between the types of semantic matches in the workflow and the number of Component Services.

4.5 Summary

In this chapter, the semantic challenge of dealing with e-services was covered. The main step for overcoming this challenge is the semantic enhancement of the functional description of the service. Such a semantically enhanced functional description is usually called a Capability Descriptions. The semantic of Capability Descriptions is generally based on ontologies. Ontologies can be described through logic languages. A Service Discovery approach using Capability Descriptions based on Description Logic was illustrated. However, such kind of approach needs expert knowledge and is time consuming in its application.

The currently favoured solutions in the context of Web Services and the semantic challenge are originated in the area of Semantic Web Services. WSMO and OWL-S define SWS languages based on XML. In Table 8 and Table 10, these approaches were classified with statements based on the four aspects of services introduces in section 2.3. Both languages demand expert knowledge in formal logics (especially DL) in order to define the meaning of Web Services [LauLar et al.2007]. OWL-S is based on ontology description language OWL, but includes some other languages for the description of capabilities (e.g. Knowledge Interchange Format, KIF). WSMO was designed from the beginning as a set of layered languages of logic, especially WSML. The ontology base of WSMO is given with its conceptual model, i.e. WSMF (Web Service Modelling Framework).

The service concept in OWL-S links the profile, service model, and grounding. The profile of a Web Service can be positioned in a hierarchy of profiles. In this way, a hierarchical categorization of services could be created. However, since the criteria for this categorization are arbitrary and not based on an agreed semantic, this hierarchization of service profiles can only be of practical use in a very restricted domain of services.

WSMO introduces besides services so-called goals, in order to support the different views on services of Service Requester and Service Provider. In OWL-S, the Service Profile is used for the Service Request and the Service Offer. OWL-S supports mediation only as a part of the underlying Web Service infrastructure. For a semantic description, WSMO relies on loose coupling with strong mediation. It uses the four elements service, goal, ontology, and mediator. WSMO includes mediators between goals, which allow the definition of goals by refining existing ones. A goal is linked with a service through another mediator type, which means flexibility for the matching (phase 3 of 4th aspect of services).

Considering the third aspect of services, WSMO explicitly defines the orchestration of the service through describing the other services or goals. WSMF, the conceptual model of WSMO, allows a more robust approach of the orchestration than OWL-S. The use of goals in the orchestration description includes descriptions of the required functionality instead pointers to the concrete Service Providers in advance. Similar to BPEL, the Service Providers are specified in the Grounding Model of OWL-S. This could lead to a drawback, when one of the chosen Service Providers is not available. However, an explicit orchestration description, especially in terms of some process languages, is easier maintainable, can exist independently of a specific requester agent, and can be passed between agents as a data structure. This approach is used to a great extent by the OWL-S virtual machine.

The second aspect of services, i.e. the non-functional Service Description is partly considered in SWS: The OWL web service ontology offers "placeholders" for the description of non-functional service properties, along with a minimal number of specific non-functional properties. In the context of the OWL-S profile, the non-functional properties of services are considered to be almost entirely domain specific. The Web Services Modelling Ontology (WSMO) uses Dublin Core metadata as the core properties, and then extends these to include some web service properties. The model is extensible and caters for domain-specific inclusions. The Web Services Description Language (WSDL) presents an entirely functional view of services and was not intended to attempt the description of the non-functional properties of services.

Concluding, it can be stated that Service-oriented architecture became the standard paradigm for software component integration. However, with the permanently increasing amount of available services and dynamic changes, the complexity of such service infrastructures, their maintenance, and consequently the expenditures spent for their operation increase equally. In order to deal with these effects, a higher degree of automation as well as categorization of approaches and ontologies became necessary. The ultimate goal is Autonomic Service-Oriented Computing discussed in the next chapter.

5 Autonomic Service-Oriented Computing

Formal representations of services and matching are required for a principal pre-selection of services. The pre-selected services have to be further tested if and on which level they satisfy a given Service Request. Subsequently, negotiations with the Service Providers are necessary, before a deployment can happen. The consumption of the service might end with a user feedback or a settlement of the involved costs. The automation of this life cycle of services is addressed through **Autonomic Service-Oriented Computing** (**Autonomic SOC**).

The goal of Autonomic SOC is a reduction of human intervention following some ideas of Autonomic Computing, which traditionally aims at automating of business processes and workflows. An enumeration of requirements for autonomic Service Composition and discovery can be found in the survey of [RamHol et al.2009]. The goal of Autonomic SOC includes the overcoming of challenges discussed in previous chapters: (1) the integration challenge (scalability, robustness, and a flexible matchmaking) as well as (2) the semantic challenge (semantically enhanced Service Description and an improved Service Discovery).

The additional challenges of Autonomic SOC are

- Dynamic changes of Service Offers and user preferences,
- Permanently increasing amount of available e-services of different domains,
- The extremely rising number of approaches with innovative algorithms in the area of (Semantic) Web Services,
- The quantity of Web Service standards and their versions,
- Mediation between different approaches and expert knowledge is often necessary for the composition of services of different domains or Service Designs.
- The inherent complexity of services and the various (implicit) policies in this context.

Autonomic SOC was introduced as a goal in the e-business domain, which was traditionally a main force for the development of SOC. Nevertheless, other domains have a similar interest in this goal and take advantage of developments in the area of (Semantic) Web Services. Requirements similar to Autonomic SOC arise for instance with the transparent use of technical facilities, which is a main concept of CSCW and Cloud Computing.

The advent of mobile computing devices as well as the development of wireless, ad-hoc networking technologies has led to the growth of infrastructure-less environments. Smart phones couple computational power for services with the ability to connect to other small devices. The research of [WeeWar2010] includes the operational context described through non-functional properties, like location, performance, power and network, in order to manage services and device access. A federated approach for Technical Services in a Wireless Mesh Network (WMN) is described in [KrKrKu2009]. Mobile environments can lie at the edges of Internet, i.e. they might be disconnected/sparsely connected to the rest of the world. These challenges of Technical Services, which are in principle the integration challenges discussed in context of VPO (Example 2), are now addressed by the research in context of Cloud Computing as for instance Distributed Web Services Discovery Middleware for Edges of Internet [HaMaKü2010]. In order to address the access to such edges of Internet, they proposed and experimentally evaluated an interoperability middleware that synergizes known techniques of DIS: P2P technology, message queuing support, and a passive distributed UDDI repository for Web Services discovery and invocation.

The above enumerated challenges of Autonomic Service-Oriented Computing appear so severe that an achievement of the ultimate goal appears to be impossible. However, further solutions for narrowed down domains and applications can and will be achieved. Semantic enhancements, Autonomic Service Discovery, and the research for more holistic concepts for the classification of e-services are current attempts of overcoming these challenges. Especially Autonomic Service Discovery relies on existing Web Service environments, in order to deal with the discovered Service Candidates. Execution Frameworks follow another idea, a complete support of the whole life cycle. They start with Service Discovery based on SWS solutions and use Web Service technologies for the deployment and execution of the services. It may be remarked that OWL-S was developed with the same idea in mind. The following sections have a closer look at some approaches in the direction of Autonomic SOC.

The survey is structured through the mentioned aspects of services:

1st aspect of services: The survey concentrates on the e-business domain, but it is not restricted through the perspective of one of the involved parties (e.g. Service Provider, Service Requester, and Service Trader).

2nd aspect of services: In Autonomic SOC, non-functional properties of services gain in importance and they are more integrated into the solutions. The approaches are not any longer mainly concentrated on the functional aspect. **Hybrid Service Discovery** environments incorporate non-functional aspects (e.g. [KvaRon et al.2005; WuRan et al.2007]).

3rd aspect of services: The bunch of policies coming with the orchestration and choreography of services is a big challenge for the Autonomic Service-Oriented Computing. It leads to the narrowing of the application domain. Some approaches integrate Workflow Management. The third aspect is for the chosen solution of ACTAS of less interest in this survey.

4th aspect of services, Service Design: The Service Description looks for concepts that go beyond the semantic enhancements of SWS. Some of these concepts like Query Languages, coming like SWS from the Semantic Web research, can be more associated with Service Matching. The new concepts for the Service Design allow a categorisation of services and the recording of users' preferences for their ranking (e.g. [GaRuRu2010]). Alternative languages like the Unified Modelling Language (UML) are used for the Service Description [SpaZis2010].

4th aspect of services, Service Discovery and Composition: Expectedly, the main effort of research of Autonomic SOC belongs in this category. Especially the Service Matching relies increasingly not only on property-based, but concept-based matching, i.e. methods of AI like for instance Information Retrieval (IR) are applied (e.g. OWLS-MX [KluFri et al.2009]). The research uses languages and concepts developed mainly in the area of Semantic Web (e.g. Query Languages, e.g. WSML-MX in the case of OWLS-MX), in order to extend the Web-based solutions of (Semantic) Web Services. Hybrid Service Discovery methods integrate, like previously mentioned, the non-functional aspects of services.

4th aspect of services, the other phases: The earlier stated Execution Frameworks propagate the support for all phases of the life cycle of a service relying on methods of SWS. The need for more adaptability led to the renaissance of MAS and the inclusion of external algorithms like the mediator of WSMO.

5.1 Improvements of Service Discovery and Composition

Semantic Web Services (SWS) semantically enhanced the functional Service Description of Web Services through annotation based on ontologies [Gri2007]. Service Trading is not directly addressed. First approaches relied on a centralized service registration with UDDI. In [AkkFar et al.2005], the World Wide Web Consortium (W3C) describes the storing of WSDL-S descriptions in UDDI registries. Several extensions were proposed and implemented, in order to extend the keyword access (UDDIe [ShaRan et al.2003]), the storage of non-functional attributes [BilSin2004; WaZhSu2004], or a semantic enhancement [AkkGoo et al.2003]. An alternative keyword-based search of service was offered with Woogle [DonHal et al.2004]. Service Discovery based on keywords, simple semantic descriptions, and rich semantic description (comparable to the DL Capability Description in section 4.2) are explored in [FeKeZa2008 section(s) 8.3-8.5]. The weak points of keyword based repositories like UDDI and Woogle became obvious, because a Service Discovery with an abstraction of services on level of keywords is in no way precise enough.

The survey of [RamHol et al.2009] categorizes the Service Discovery approaches through two dimensions: (1) the federation of the trading (centralized, distributed, and decentralized) (4th aspect, phase 2), and (2) the Service Description (syntactical, hybrid, semantically enhanced) (4th aspect, phase 1). It evaluates to which extent existing Service Discovery frameworks fulfil the criteria of Autonomic Service Discovery. The criteria used for the classification of the approaches in [RamHol et al.2009], Service Trading and Service Description, belong to the phase 1 and phase 2 respectively of the fourth aspect, introduced in section 2.3.

Besides Service Trading, the Service Discovery is based on Service Matching. The matching of a Service Request with a Service Description of a Service Offer can involve several aspects of services. A (structural) functional matching checks if the arity of the methods and the types of their input and output parameters fit together. Many approaches ([PaoKaw et al.2002; PeNiHu2009]) consider several degrees of matching for the types (for instance: exact, plug-in, subsumes, and fail). Alternatively, graphs can be used for matching like UML class-diagrams for the functional matching ([HaReMa2004; SpaZis2010]). In some approaches [SpaZis2010], the names of the methods are compared in a linguistic way. Capability matching of semantically enhanced notations also includes Preconditions and Effects (e.g. IOPE capability on page 42). WSMO introduces the “goal” element for the requester´s desires, which has to match with the capability of a Web Service definition of a Service Offer. The inherent complexity of a service is considered through the behavioural functional matching which takes advantage of BPEL or WSCL descriptions. Non-functional attributes, like Quality-of-Service (QoS) and calculated similarity distances, are in particular used for the selection and ranking of found Service Offers. Additional constraints (hard and soft) are also applied for the selection of services.

The mediation of WSMO allows an adaptable Service Matching, i.e. the use of Service Matching algorithms on data, which is defined in distinct semantic contexts or formalisms. Some proposed standards of SWS extend the Service Matching to the methods of AI. The DFKI[4], proposes a Service Matching approach for various SWS standards (OWLS-MX [KlFrSy2006; KluFri et al.2009], WSMO-MX [KluKau2009], and SAWSDL-MX [KlKaZi2009]), which enhance the property-based Service Matching through methods of the IR (Information Retrieval). In the internet document [Klu2008], Matthias Klusch compares several Service Matching approaches. Service Matching approaches rely often on Query Languages (e.g. iSPARQL [KiBeSt2007] for the approach OWLS-iMatcher). Query Languages belong to the research area of Semantic Web. Nevertheless, they are also used for the investigation of repositories of Semantic Web Services, in order to find matching services based on the similarity of their properties/data. There have been proposals for query languages to support Web Services Discovery. In [BeeEya et al.2006], the authors propose BP-QL a visual query language for business processes expressed in BPEL. The Unified Service Query Language (USQL), an XML-based language to represent syntactic, semantic, and Quality-of-Service search criteria is described in [PanTsa2009]. An extension of USQL that incorporates a behavioural part has been proposed. The support of the design process (phase 1) of service-based systems is also a possible application of Query Languages.

Several techniques have been proposed for Web Services Composition, many based on AI planning (e.g. [PisBer et al.2004]) or on logical deduction like Linear Logic (LL) in [RaKuMa2004]. The AI planning-based Service Composition uses methods of AI for generating a plan for composition before the actual Service Composition is performed. Improving the accessibility of cloud services for non-computing experts is the concern of [BroGos2010]. It promise to ease the discovery, selection, and use of clusters within a cloud.

The checking and ranking phase (phase 4 of the life cycle in Fig. 2) is addressed in [GaRuRu2010]. Service Discovery and especially Service Ranking is often based on users' preferences. The description model of these preferences is many times ad-hoc and depends on the discovery framework, the domain, and the context of the environment. In order to ease the Service Ranking, [GaRuRu2010] figures out a lack of a general, comprehensive, and user-friendly preference model, which has to be overcome. The proposed model is based on query preference model known from database systems. A concrete implementation of the model in WSMO is outlined.

[4] http://www.dfki.de/web, Deutsches Forschungszentrum für KI, German Research Centre for AI

5.2 Semantic Web Services Execution Frameworks

For achieving the goals of Autonomic SOC, several initiatives exist to create comprehensive frameworks that integrate the vision of SOA and SWS (METEOR-S, IRS, and SESA). In a semantics-enabled world, the coordination between systems is executed through the use of well (semantically) described services. The services have, according to the discussed life cycle of a service (cf. section 2.3.4), to be discovered and selected on the basis of requirements, then orchestrated and adapted or integrated.

5.2.1 METEOR-S

SAWSDL is part of the METEOR-S project [VerGom et al.2005], which wants to extend semantically the basic standards of Web Services. In this way, METEOR-S supports the phases of the life cycle of a service (4th aspect of services). METEOR-S Web Service Discovery Infrastructure (MWSDI [Kaa2003a]) incorporates semantic into UDDI with the t-models, in order to have firstly separate registries associated with domains, and secondly an own Domain Ontology for each of these registries. METEOR-S supports mediation and the Service Composition of BPEL. The latter is part of the METEOR-S Web Service Composition Framework (MWSCF [Kaa2003b]), which makes use of semantic process templates. The executable BPEL representation is generated with an explicit process dataflow of found concrete Web Services. The Service Discovery of METEOR-S includes the domains in a static way, i.e. through the use of separated registries.

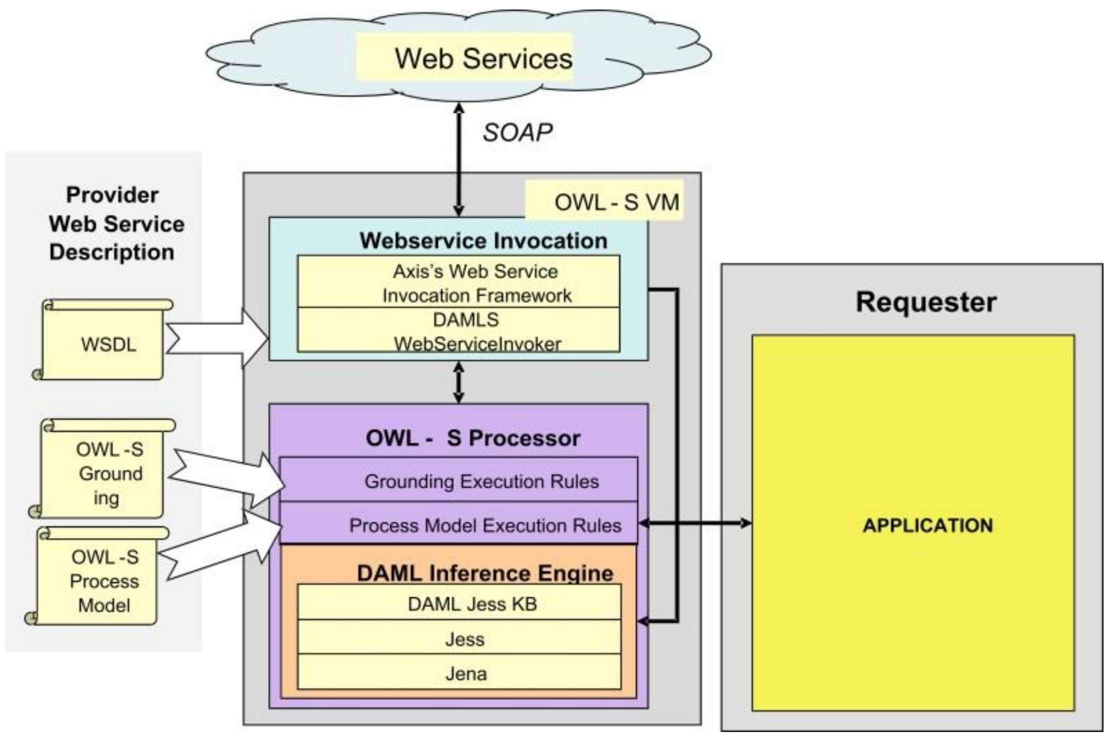


Fig. 16 - OWL-S Virtual Machine (VM) [Pao2003]

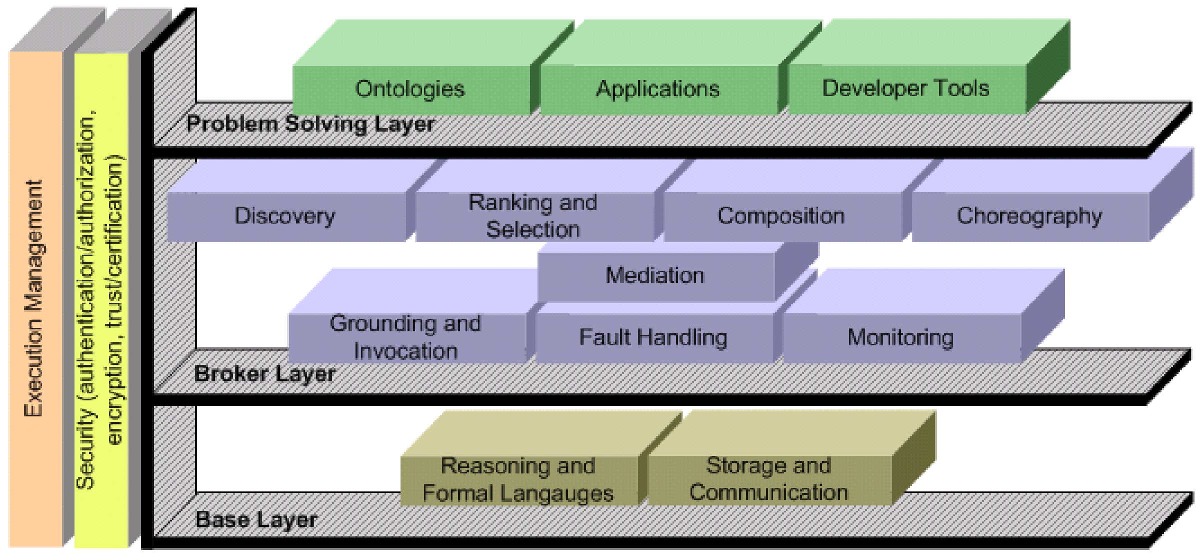


Fig. 17 - SESA

5.2.2 IRS

IRS III [DomCab et al.2004] is a framework and implemented infrastructure which supports the creation of semantic Web Services according to the WSMO ontology. IRS III has four main classes of features which distinguish it from other work on semantic Web Services. Firstly, it supports one-click publishing of 'standard' programming code. In other words, it automatically transforms programming code (currently it supports Java and Lisp environments) into a Web Service, by automatically creating the appropriate wrapper. Hence, it is very easy to make existing standalone software available on the net, as Web Services. Secondly, by extending the WSMO goal and Web Service concepts users of IRS III directly invoke Web Services via goals, i.e. IRS III supports capability-driven service execution. Thirdly, IRS III is programmable. IRS III users can substitute their own semantic Web Services for some of the main IRS III components. Finally, IRS III services are Web Service compatible – standard Web Services can be trivially published through the IRS III repository and any IRS III service automatically appears as a standard Web Service to other Web Service infrastructures.

5.2.3 SESA

The global architecture of **Semantically Enabled Service-Oriented Architectures** (**SESA**) [FeKeZa2008] comprises several layers between so-called stakeholders and the Service Providers: (1) stakeholders forming several groups of users of the architecture, (2) problem-solving layer building the interface for stakeholder access, (3) Service Requesters out of the problem-solving layer, (4) Middleware providing algorithms for the integration and interoperation of services, and (5) Service Providers offering the services. The middleware consists of a Broker Layer and a Base Layer (Fig. 17). The Broker Layer holds algorithms for discovery, selection, negotiation, composition, choreography, mediation, grounding, fault handling, and monitoring. The Base Layer provides exchange formalism as well as resources for storage and communication. The architecture comprises layers, which were introduced in section 3.3 (Fig. 9) as horizontal layers, which are besides other functions responsible for execution management and security. SESA is based on WSMO and its Web Service Modelling Language (WSML). A reference implementation of SESA was done with the Web Service Execution Environment (WSMX).

5.3 Agents and Web Services

Service-Oriented Computing can benefit from Multi-Agents system technologies by adopting the coordination mechanisms, interaction protocols, and decision-making tools designed for Multi-Agent systems. JIAC V [DAI2008] is a MAS middleware following this idea. The publication of [PoToTu2007] speaks of agent-based SOA, the integration of the agent technology with other strategic technologies like Web Services, workflow, rule engine and semantic Web. The paper of [BroUro et al.2009] demonstrates the use of a decentralised Multi-Agent system, in order to support the discovery, selection, and negotiation of services. MAS frameworks used for the Service Composition incorporating negotiation can be found already in [PreByd et al.2001], which introduces an algorithm for Service Composition through negotiation with multiple auctions, in order to meet the needs of the Service Clients. Dickinson [DicWoo2005] argues that agents and Web Services are distinct. Agents provide a distinctive additional capability in mediating user goals to determine service invocations. The paper of [DicWoo2005] illustrates one approach using reactive planning to control web-service invocation by BDI agents.

In [MahSpa2010], researchers of the City University introduce a MAS framework supporting the SLA negotiation for service-oriented systems. Combined with dynamic Service Discovery, the paper shows, how MAS frameworks can propagate the vision of Autonomic SOC. Candidate services are recognized, which can be used for a Service Composition. The agreed but not enforced SLA is guaranteed for a certain period, in order to ensure the availability of the services in the Deployment phase. In this way, the approach supports non-functional aspects of availability.

A MAS based framework, which supports the Service Composition by non-IT-experts, is MAMS introduced in [ThiKon et al.2009]. MAMS is an application of the agent middleware JIAC V [DAI2008] (cf. section 3.2). The MAMS service framework provides an infrastructure for the creation, deployment and execution of Service Compositions. It offers a graphical service creation environment like WfMs (cf. page 30) as well as a service execution platform based on intelligent agents. A service is represented through an agent. MAMS claims the improvement of scalability, management, and stability. Service Matching, runtime load balancing, and self-healing mechanisms could be tested with this environment.

A framework based on a Multi-Agent system made up with agents occupying five distinct roles, namely a Planning agent, an Execution agent, a Composition agent, a Discovery agent, and a Monitoring agent, is the approach introduced in [ChMeGh2010]. Obviously the approach tries to support the whole life cycle of a service, in order to achieve Autonomic SOC. However, it can be doubt that the variety and the inherent complexity of Composite Services in the different domains, which also incooperates policies and business processes, can be covered with agents, especially when one looks at the execution and monitoring of services.

5.4 Artificial Intelligence (AI) in SOC

The Service Description in an alternative language, i.e. UML, shows an approach of the City University [SpaZis2010]. It is an UML-based framework in the domain of Software Engineering, which looks at the composition and integration of software systems composed of autonomous services and other mostly locally available software code. The idea of autonomous services is similar to autonomous agents (cf. Fig. 6 and section 2.3.1). In order to support the development of these systems, it is necessary to have new methods, processes, and tools. The framework adopts an iterative process in which software services that can provide functional and non-functional characteristics of a system being developed are discovered, and the identified services are used to reformulate the design models of the system. The framework uses a query language to represent structural, behavioural, and quality characteristics of services to be identified, and a query processor to match the queries against service registries. The matching process is based on distance measurements between the queries and service specifications. A prototype tool has been implemented. The work has been evaluated in terms of recall, precision, and performance measurements.

A subsection of the domain information retrieval using artificial intelligence is data mining. A special field of data mining became the recorded access pattern of Web logs. This information can be processed and compressed in so-called Web Access Pattern trees (WAP trees). PLWAP algorithm uses a preorder-linked, position coded version of WAP tree and eliminates the need to recursively re-construct intermediate WAP trees during sequential mining as done by WAP tree technique. The approach of [WanTsa et al.2010] extends these data mining methods to Composite Service Discovery. They utilize a PLWAP-tree algorithm to analyse the relationship among Web Services from Web Service usage log. In this way, concerning the introduced aspects of services, this approach combines methods of the matching phase with special algorithms for the trading phase. They generated time-ordered sets of Web Services, which could be exploited, in order to integrate them into a real business process. The papers shows, that according to mining results Web Services can be integrated into a Composite Service with Service Composition based on process level, i.e. a dynamic orchestration.

5.5 Enhancements through new concepts and algorithms

Enterprise Application Integration (EAI) can dynamically deal with the integration of applications due to the use of Message Brokers (cf. section 3.1). [PaDaDi2010] introduces Service Application Integration (SAI) with message-based Service Brokering and dynamic Service Composition for a loose coupling between Service Providers and Service Clients. In this way, the publishing and discovery of services, the base of SOA, would not be any longer a prerequisite. The dynamic Service Composition of SAI is based on an Artificial Intelligence (AI) planning approach and on the adoption of an ontology-based functional profile encoding information for enabling automatic information extraction and combination in the Service Composition chain.

Dynamic service reconfiguration and automated enactment is also topic of [SplBra et al.2009]. The publication speaks of Open Matching Architecture, i.e. the matching of services is not limited to a pre-determined set of matchers and repositories. The proposed architecture consists of three, previously developed, components: the CoWS template-based reconfiguration service, the Knoogle MatchMaker service, and the Triana workflow enactment engine.

The publication of [SaNaMa2006] points out that the solution of the semantic challenge should consider the context of the composition and execution of Web Services. Needs, preferences, and Service Capabilities vary over time. Contextual details allow a kind of categorization of these service characteristics for an improved tracking, bringing Web Service advertisements and user requests together. The approach of [NoSaZa2007] develops ontologies for preferences and capabilities expressed with SAWSDL. [MicChi et al.2007] discuss a context-based mediation approach, in order to solve semantic heterogeneities between composed Web Services.

In the project ConWeSc (Context-based Semantic Web Services Composition) [SaNaMa2005], an alternative approach to OWL-S namely OWL-C (Ontology Web Language-based Context Ontology) for context aware SWS was developed. Similar to the challenges of EAI, the composition of Web Services, originated from different Service Providers, has to mediate between various service contexts. For the semantic challenge, the Service Providers will agree on an appropriate ontology. For the coordination and the integration challenge, the context of the Service Providers, for instance the local time, is important, in order to avoid conflicts. [SaNaMa2006] introduces a typing of Web Services and constraints like maximum number of available Web Service instances. The approach distinguishes between Composite Service, Abstract Service (simply addressed as Web Services), and Concrete Services (addressed as service instances). Accordingly, a context can be declared with a C-context, W-context, or I-context type, respectively. Based on these service contexts, fitting security contexts (CSec/WSec/ISec-contexts) can be derived.

The paper [StAlJo2008] mentions that context-awareness is highly desired across several application domains. SWS technology supports the automatic allocation of resources for a given well-defined task. However, it does not entail the discovery of appropriate SWS representations for a given situational context. A situational context depends on the domain, its complete notion in all its facets is (too) costly, and real-world situations show a too big variance. The publication proposes a model derived from the idea of Conceptual Spaces: Conceptual Situation Spaces (CSS). CSS is aligned to established SWS standards and enables the description of situation characteristics as member in geometrical vector spaces. Semantic similarity between situations is calculated in terms of Euclidean distances between CSS notions. The approach extends SWS descriptions with the context information of CSS. In this way, the matchmaking can include the similarity-based real-world situation characteristics.

5.6 Summary

Semantic technologies can facilitate the integration of services by means of semantic Service Descriptions and artificial intelligence methods. On the one hand, it can be argued that a burden for service processing and performance araises from the complexity of semantic languages as well as integration techniques that depend on logical reasoning. On the other hand, the difficulties of management of Service Descriptions rise dramatically with their complexity, when there is no autonomic control. The logical reasoning can efficiently help to resolve inconsistencies in Service Descriptions as well as maintain interoperability, when these descriptions change. Therefore, SWS seems to open up the way for Autonomic SOC, the vision of providing services transparently like electricity today. Cloud Computing is following this idea, but it still has to adulate from Software Distribution ideas towards the complete Autonomic SOC with extended service integration and an support of services as a software paradigm.

6 Problem Statement

The State-of-the-Art discussed the existence of many approaches for overcoming of challenges occurring with integration and diverse semantics of e-services. These approaches are based on their proprietary methods and repositories, which are often domain specific. However, Autonomic SOC means that services of different domains and service environments should be discovered and become composable with a minimized involvement of human beings. Thus, the Service Discovery has to integrate autonomic mediation for properties of various Service Descriptions, in order to be adaptable to different interfaces and ontologies. It will be a future challenge to provide appropriate algorithms and ontologies, in order to ease the Service Discovery and Service Composition. A problem for the Autonomic SOC will be always the inherent complexity of services (3rd aspect of services) originated often from proprietary policies. The Lock-in-Effect of Cloud Computing is an example in this direction. Cloud Computing is also an example for the integration of services of technical and business domains. In summary, it appears that Autonomic SOC has to adopt a framework character, in order to integrate existing Service Discovery environments.

ACTAS – ADAPTIVE COMPOSITION

7 Hypothesis of ACTAS

Overview

The application of ACTAS …

1. **reduces the effort for the discovery and composition of services,**
2. **takes advantage of existing algorithms for dealing with Service Description, Service Matching, and Service Mediation,**
3. **allows the definition of Service Composition,**
4. **adapts to the policies of the parties involved in SOC,**
5. **supports availability control.**

The State of the Art showed a variety of approaches of Service Discovery, Service Composition, and Service Matching introduced for the overcoming of the challenges of SOC and developed with the ambitious goal of Autonomic Service-Oriented Computing. In the presentation, the approaches were partly classified through the presented four aspects of services. In this chapter, the hypothesis of ACTAS [BeKlMe2000; KlHoSc2000; KlReSc2002; KlUnBr2009, KlUnBr2010] – Adaptive Composition and Trading based on Agents - is introduced and its points shortly illustrated. ACTAS is a framework defining three models for services, requests, and a declarative Composition Process. The goal of ACTAS is the discovery and composition of multiple solutions of SOC with the consideration of their semantic context.

The goals of Service Requesters have to be compared with the Service Offers. The Service Matching is about finding common elements in the descriptions. Depending on the level of detail, in which these entities are considered, the models of services can be described at varying levels of abstraction. At the most fine-grained level in the aspect of the functional description of a service, services can be seen as concrete state transitions from a pre-state to a post-state (the Capability Descriptions of SWS). On a more abstract level, services can be understood as abstract objects characterized through their properties. On this level of abstraction, services can be considered as instances of ontological concepts. A Service Description describes such a concept traditionally from its functional aspect.

It is the idea of ACTAS to use concepts for the classification of services (Semantic Characteristics), which describe the services with various aspects. Some of these concepts are used for the discovery of services and for the description of compatibility (Compatibility Characteristics). The classifications are semantically described through relationships with criteria originated on the four aspects of services introduced in section 2.3. The hypothesis of a reduced effort of Service Discovery and Service Composition (bulletin 1 of hypothesis) is based on the assumption that services belonging to the same intersection of categories are more likely to be compatible. Such services are called principally compatible.

In a subsequent step, constraints working on the level of Service Properties will check the selected services closer through approved algorithms fitting to the categories of these Candidate Services (bulletin 2 of hypothesis). First of all, each Service Property will be examined, if its information fits to the constraints valid in the semantic context of the categories of a Candidate Service (Value Constraints). Secondly, the compatibility will be tried on the level of the Service Property, i.e. the information of the Service Properties of two principally compatible services is tested for matching (Merge Constraints). Finally, constraints are checked for the information held in several Service Properties of the Composite Service (Exchange Constraints). The last checking can realize a mediation of information.

The categories for Service Requests are closer defined with relationships to ontological concepts for the classification of user groups, in order to declare which kind of Service Clients shall have access to the services belonging to these categories. In this way, categories for the description of B2C interfaces for specific groups of Service Clients can be introduced (bulletin 3 of hypothesis).

The Multi-Agents System of ACTAS provides software agents acting pro-actively complying with the policies of the parties involved in SOC: Service Provider, Service Requester, and Service Trader (bulletin 4 of hypothesis). ACTAS introduces the additional role of a Service Client for the support of Technical Services. The Composition Process is initiated from the user application. Therefore, it can be performed by agents, with a pro-active behaviour adapted to the application.

The non-functional criterion "availability of services" plays a special role in the selection of Service Candidates. On the one hand, categories of services could be defined; whose members support a certain availability control algorithm. On the other hand, selection of any Service Candidate is senseless, if the service is not any longer available in the Deployment Phase, due to the fact that the Component Service and its resources were not reserved. Therefore, availability control has to be part of Service Discovery in Autonomic Service-Oriented Computing. The data model of ACTAS supports availability control through the agent of the Service Provider (bulletin 5 of hypothesis).

The goal of the thesis is the proof of the feasibility of ACTAS as a framework for the discovery and composition of services. Data models for the keeping of informal data of the Service Description and the Service Request were defined: Service Model (S-Model) and Request Model (R-Model). A third model, the Composition Model (C-Model) realises the Composition Process with a declarative environment. It is possible to translate the hierarchical data structures of the S-Model and R-Model into e.g. XML format, in order to transfer their information over the Internet. The C-Model should incorporated handles to the access of implementation instances of algorithms associated with the Service Properties. The C-Model and the scope of the thesis are restricted to the Composition Process. The assumption is made that fitting algorithms can be integrated into the declarative environment as for instance PROLOG modules. Other implementation instances of the algorithms could be accessible through (statefull) Web Services. It is not the goal of the thesis to implement these algorithms, their interfaces, or the environment of the Multi-Agents Systems.

8 System Environment

ACTAS uses a Multi-Agent system providing these kinds of agents (cf. Fig. 18): Request Agent (ReA), Facility Agent (FA), Trader Agent (TrA), and Composition Agent (CoA). Principally, the MAS environment also includes a Personal Agent (PA) representing an application user, i.e. a potential Service Client. The ReA realizes the interface to the user application. The FA fulfils several tasks: (1) the publishing of the Service Offer of a Service Provider, (2) the availability control of the Service Modes, (3) a potential reservation of a selected Service Mode, (4) a resource management., (5) the negotiation with other Facility Agents about their found Component Services, and (6) the deployment of the service, when the service is not an Abstract Service. In the case of an Abstract Service further orchestration will take place in the Service Grounding phase after the processing of ACTAS (cf. section 8.5). (A principal Sequence Diagram is in Fig. 35 at the beginning of the description of the R-Model.)

ACTAS assumes the generation of the Service Request inside of the Application Environment leading to the addressing of the ReA. Subsequently, the ReA creates a CoA for this request. . In a simple case, a web browser together with an add-on could take over the role of a ReA with its Application Environment. In a successful Service Discovery, the CoA will come back with a set of Service Candidates. Since the ReA is related to a defined application, the algorithm of the CoA can be adapted to this application. Thus, a CoA could for instance deliver exactly one matching service or return multiple candidates, depending on the policy of the application. In the latter case, it is up to the negotiation phase to select a service for the Service Level Agreement (SLA). The composition task of the CoA is comparable with the composition of concurrent processes. In comparison to the methods of coordinating concurrent processes, the principal methods of composition can be discussed.

In some MAS based approaches (e.g. [KünMat2006; MüKoBr2006]), each (component) Service Candidate is represented through an agent, in order to let them actively test the compatibility among each other. In ACTAS, the differentiation of the agents is oriented at the distinct phases of the life cycle and the involved user roles. The pro-active behaviour of federated trader agents forms the trading phase. The CoA supports the Service Matching and Service Selection. Personal Agents and Facility Agents include user roles.

The System Environment introduces several user roles: ACTAS Administrator, Service Administrator, Service Designer, Service Provider, Service Requester, and Service Client. The

Request Agent takes over the role of the Service Requester. ACTAS as a framework for the automation of the Service Discovery introduces Semantic Characteristics and Property Classes as new entities. It is the task of ACTAS Administrator to provide a semantic classification and a publication for these entities.

The Service Provider operates the FA, in order to publish and provide his services. A Service Provider can take over several roles: on the one hand, he will compose Service Descriptions as a Service Designer using the ACTAS entities, which might include a possible automatic adaption of the Service Descriptions. On the other hand, the services and their resources have to be managed in a role as a Service Administrator. It is a basic idea of ACTAS that only a Service Administrator/Designer, responsible for a certain family of services, can describe properly a

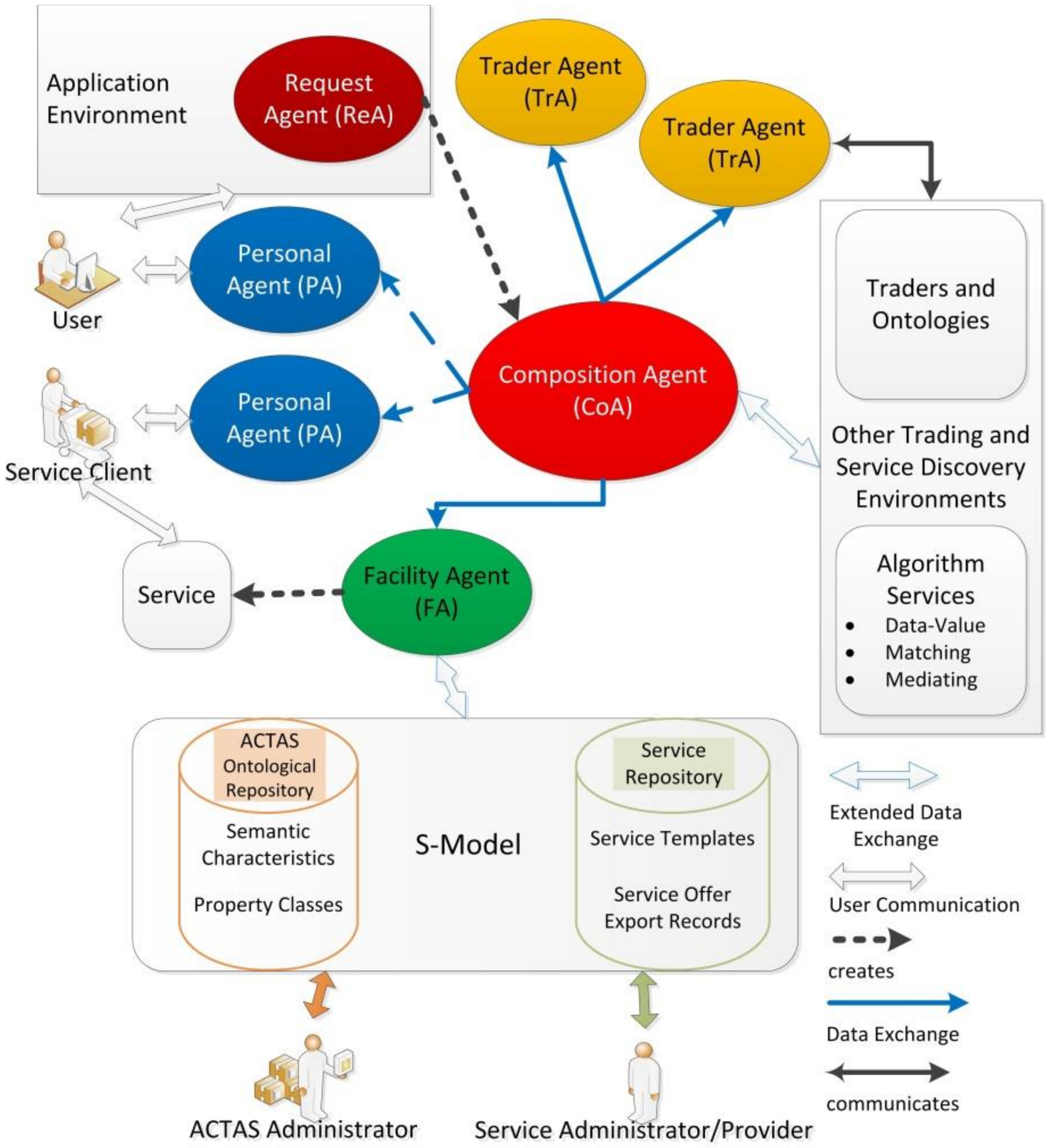


Fig. 18 - Agents Environment of ACTAS

service with its properties, its compatibilities, and availability. However, many Service Discovery approaches for example in (Semantic) Web Services take a kind of "God-view", i.e. the services have to use the given formalism of the Service Description lacking of adaptation to other kind of service (e.g. technical ones), and of the support of availability control. ACTAS distinguishes between Service Templates and Service Offers. Additionally, the new entities of ACTAS allow an adaptation of the semantic context and of the used algorithms. The new entities also allow an adaptation of the Service Request to a fixed group of Service Requesters/Clients.

Web Services describe only two user roles: Service Requester and Service Provider. This is due to the simple scenario of SOA, where the Service Requester will also become the Service Client later on. ACTAS introduces the additional role of Service Clients, in order to provide a more general framework for the Service Discovery, where a service can have several Service Clients (e.g. a Communication Service).

In Fig. 19 and Table 11 an overview of the life cycle of services together with the MAS of ACTAS is given. In the following sections, the System environment is described more in detail. The symbol of ACTAS in the schematic figures in these sections needs an elucidation. First of all, it illustrates the involvement of ACTAS. Secondly, several agents of ACTAS (FA, TrA) can have besides the main declarative environment of the CoA their own declarative environment of ACTAS, in order to deal with the Service Offers and to perform an extended checking of matching (cf. section 8.3). Therefore, it is symbolized that the agent communication goes

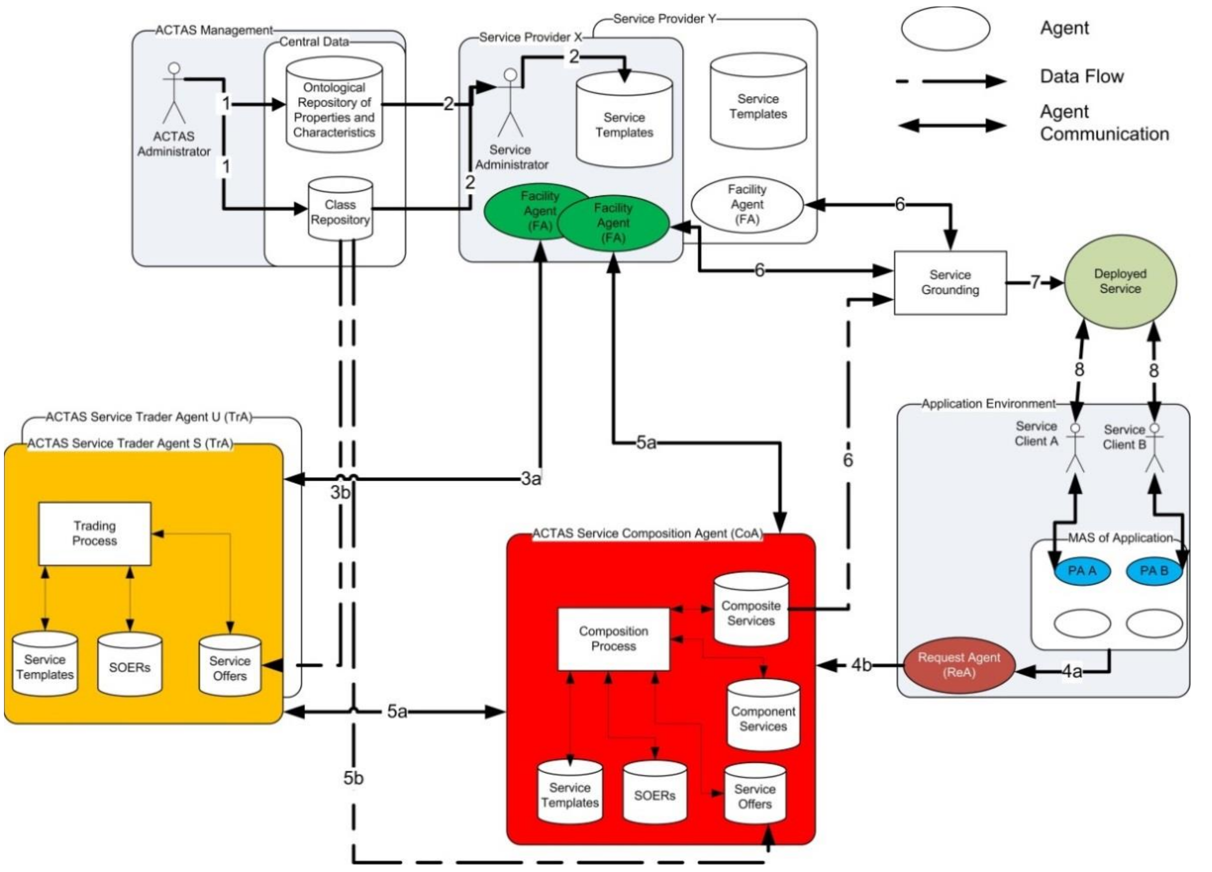


Fig. 19 - MAS of ACTAS and life cycle of Service

principally through the interpretation of the federated declarative environments of ACTAS in the first four phases. The Service Offers construct properties and constraints with the Property Classes, i.e. the declarative environment has to provide an interface to the referenced external algorithms for data-setting, matching, and mediation. This is also symbolized.

Phase	Description
Phase 1 – Service Design	• In the Service Design, the ACTAS Administrators create and publish the new entities of ACTAS ((1) in Fig. 19): Property Classes and Semantic Characteristics. The publication of these entities is done through ontological repositories, in order to achieve a commonly agreed semantic. • Service Providers build Service Templates with these elements ((2) in Fig. 19). • The Facility Agent (FA) publishes Service Templates and Service Offer Export Records (SOER).
Phase 2 – Service Trading	• ACTAS distinguishes between Service Templates (ST) and Service Offers (SO). The agents use ST and SOER for the construction of a Service Offer (SO). • The federated Trader Agents (TrA) select and compose services with both ST and SO according their policies ((3) in Fig. 19).
Phase 3 – Service Matching	• In the beginning of the Matching Phase (phase 3), the Service Request Agent (ReA) receives the Service Request (SRe) of the application environment ((4a) in Fig. 19). • ReA creates for the processing of the Service Request a Composition Agent (CoA) ((4b) in Fig. 19). • The CoA communicates with the Facility Agents and Trader Agents, in order to find principally compatible Service Offers, described with the same kind of characteristics ((5a) in Fig. 19). • The CoA uses the found SOER, its ST, the Property Classes of the referenced Characteristics and the Value Constraints of the Service Description for creation and initialization of Service Offer (SO) with the principally compatible Service Mode ((5b) in Fig. 19).
Phase 4 – Service Checking	• Enhanced constraints of the services have to be checked: • So-called comparable Service Properties are checked with the Merge Constraints, i.e. it is tested if the properties can be matched with established matching algorithms including a potential necessary mediation. In communication with the FA or PA, the availability or acceptance of the service and its resources can be clarified. • Constraints between the Service Properties, so-called Exchange Constraints, have to be checked.

Phase	Description
Phase 5 – Service Grounding	• The oncoming phases are out of control of ACTAS, which might start a Service Discovery process on their own depending on the orchestration. However, the information gained in earlier phases can be used for the Negotiation and Grounding Phase (Phase 5). The Facility Agents can play an active role in the Negotiation Phase with the Service Requester. The Negotiation Phase concludes with the Service Level Agreement (SLA), which is a premise for the Service Grounding ((6) in Fig. 19).
Phase 6 – Service Execution	• Service Deployment ((7) in Fig. 19), the execution of the service ((8) in Fig. 19), and a potential feedback is the last phase. Feedback could be used for learning of users' preferences. Actor Service Templates (AST) could be introduced to the models of ACTAS for keeping the users' preferences.

Table 11 - Overview of the life cycle

8.1 Phase 1 – Composition Model and Service Description

In the first phase, ACTAS Administrators describe and publish the entities of the S-Model of ACTAS: Property Classes as well as Semantic Characteristics. The entities have semantic descriptions and relations between each other. The Semantic Characteristics have properties and constraints declared with the Property Classes. The Service Designer will design Service Descriptions with the "building blocks" of ACTAS. Several alternative Service Descriptions of a service are hold in Service Templates as Service Modes. The Service Administrator considers

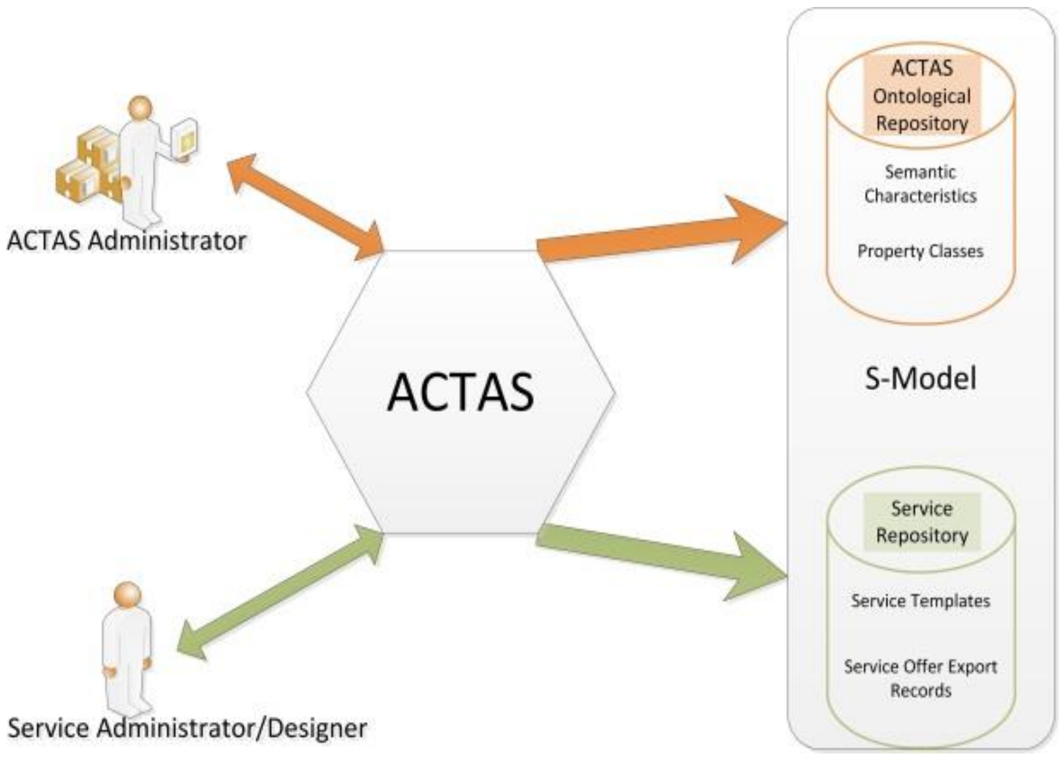


Fig. 20 - Phase 1

which Service Modes are really offered and which Value Constraints for the properties they currently have. For the publication of these considerations, so-called Service Offer Export Records (SOER) are used.

In order to consider the availability of a service the model distinguishes between Service Templates and Service Offers. The Service Provider controls the Service Description and its availability through the Facility Agents (FAs). The FA advertises the Service Template in the second phase. When a service becomes available, it publishes and manages Service Offer Export Record (SOER) accordingly. A SOER has a unique identity; and the Service Repository of a FA contains the Service Templates and current Service Offer Export Records (SOER). The ontological repositories of the ACTAS entities allow the building of Service Offers from the Service Templates. It is the task of the Service Administrator to automate the publication of the SOER through a Facility Agent. Due to the ontological semantic description of the ACTAS entities, the Service Administrator can also reflect on an automatic adaptation of its Service Templates, e.g. a new Service Mode could be automatically added; when the FA recognized that the service could be also offered with currently requested Semantic Characteristics.

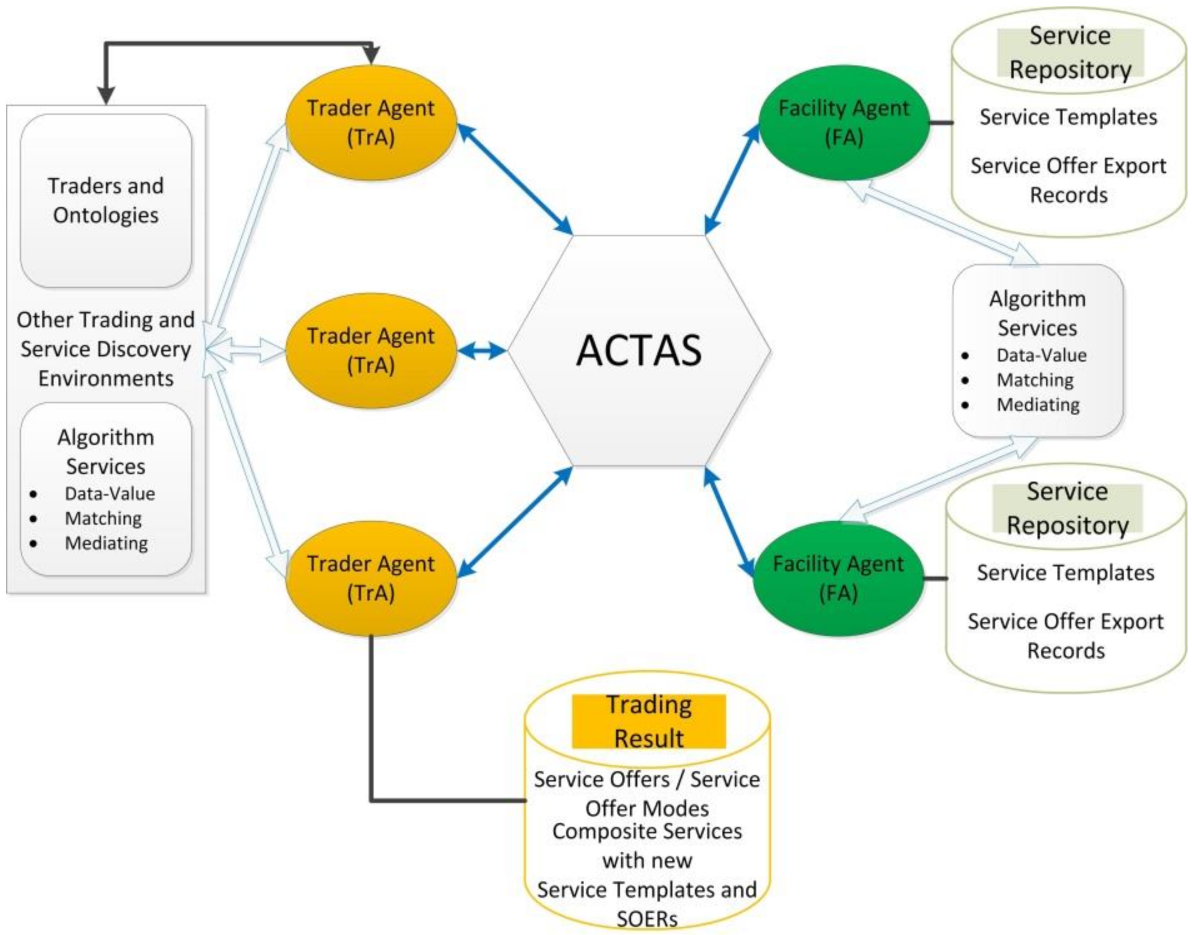


Fig. 21 - Phase2

8.2 Phase 2 – Service Availability and Trading

The Trader Agents can fulfil at least two tasks. On the one hand, they can follow their own composition policy, in order to collect or compose services of potential interest for a given application domain. On the other hand, the Trader Agents can integrate the access to external trading environments of other approaches of autonomic Service Discovery. In case of composition, the TrA can become the FA of the found Composite Service.

The Trader Agents use the SOER for the creation of a Service Offer based on its Service Template. Service Offers based on Service Templates fitting to the policy of the TrA are continuously collected and possibly composed to new Composite Services (Fig. 21). The Trader Agents (TrA) could also pre-select Service Templates without looking for valid SOERs, in order to be more adaptive with a greater selection of potential Service Candidates. An example for such a policy in COR (cf. Example 2) could be the pre-selection of Service Descriptions of communication facilities, which are available for team members of a certain VPO. However, the traders only deliver matching Service Offers to an actual Trading Request, i.e. they answer only with currently valid SOERs to Trading Requests.

A Trading Request is firstly tested on principal compatibility, i.e. a TrA checks if a Service Offer exists with an interface holding the same set of Semantic Characteristics like the Trading Request. Semantic Characteristics used for the checking of the principal compatibility are called **Compatibility Characteristic**s. Secondly, the TrA could check the matching of values of comparable properties between principally compatible Service Offer and Trading Request. The Property Classes enable the use of established algorithms for the data setting, matching, and mediation.

8.3 Phase 3 – Service Request and Composition

The **Request Agent** (**ReA**) is part of the application environment. When the need of a service arises in the application, the ReA starts the Composition Process of ACTAS. The Composition Process is described through the Composition Model or short C-Model. The Composition Process uses the data, which was collected and managed in the preceding phases.

The Service Requester, i.e. the application together with the ReA, generates the Service Request, which can be on behalf of several Service Clients. A communication service (cf. Example 1) is an example for such a case. Consequently, a Service Request in ACTAS can consist of several Client Service Requests. Service Request describes the desired service from the view(s) of the Service Client(s). Therefore, the model introduces a special kind of Compatibility Characteristics, which are called **Request Characteristic**s. Finally, the Request Agent creates a **Composition Agent** (**CoA**), which is responsible for the generation of a so-called Composite Structure.

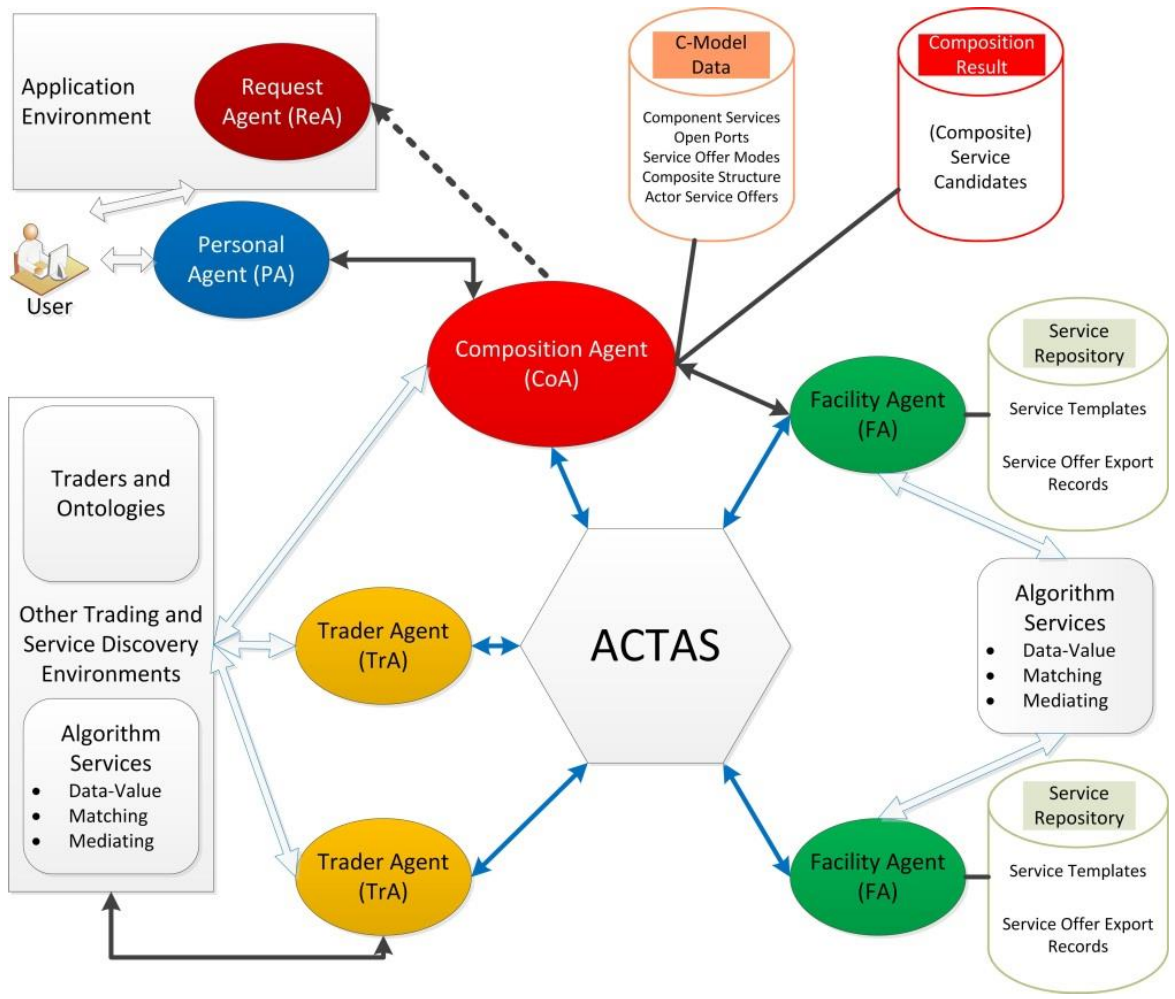


Fig. 22 - Phase 3

The CoA composes the Composite Service through the building of Service Offers using the Data-Value algorithms given with the Property Classes. The initial data structure built from the Service Request is an Actor Service Offer (ASO) for each Client Server Request. Thus, the Client Server Requests and therefore the whole Service Request become data structures dual to the Service Offers. The general policy of the CoA is to find principally compatible Service Offer to the ones, which are already part of the Composite Structure (including ASOs). Two Service Offers are principally compatible if both have a Service Port as interface holding the same set of Compatibility Characteristics. This principal compatibility idea is equal to the one discussed in the context of a Trading Request (cf. section 8.2). This means, the CoA will look for Service Offers as long as open Service Ports exist. For an enhanced matching of Service Offers, further constraints have to be checked in phase 4. The principal compatibility leads to a selection of a Service Mode. The result of the third phase will be a number of Service Candidates.

In the Service Discovery, the Facility Agents and Trader Agents are actively involved. ACTAS does not have a central repository. Each FA decides if it can offer a service holding the requested characteristics. Normally, this is done with the Service Template. However, since the entities of ACTAS are hold in an ontological repository having a semantic description, a new service mode could be dynamically generated. It is up to the publishing FA and TrA, if they extend the test of the principal compatibility to the checking of the matching of comparable properties. In the terms of ACTAS, this means the agents could additionally check Merge Constraints in the context of their declarative environment. The next phase will include the checking of Merge Constraints in the context of the declarative environment of the CoA, i.e. the data of the whole composite structure and its domain ontologies (cf. section 4.1) can be involved.

8.4 Phase 4 – Checking constraints

The Composition Result of the phase 3 was a first collection of Service Candidates. In a simple case, a Service Candidate is a Service Mode of a Service Offers, which is principally compatible with a Service Request (more precisely a Client Request of the Service Request). In a more complex case, the principally compatible Service Candidate is a Composite Service with also principally compatible Component Services.

In the phase 4 of the Service life cycle, constraints for the selection of matching Service Candidates are performed. Firstly, the matching auf the Component Services and especially the matching of the Service Offer with the Service Request are tested. This is done through the so-called Merge Constraints. The interface of a (Component) Service, i.e. of a Service Mode of a Service Offer, is its Service Port. Since a Service Request and a Candidate Service as well as the Component Services of the Candidate Service are called principally compatible, their Service Ports must hold the same sets of Compatibility Characteristics.

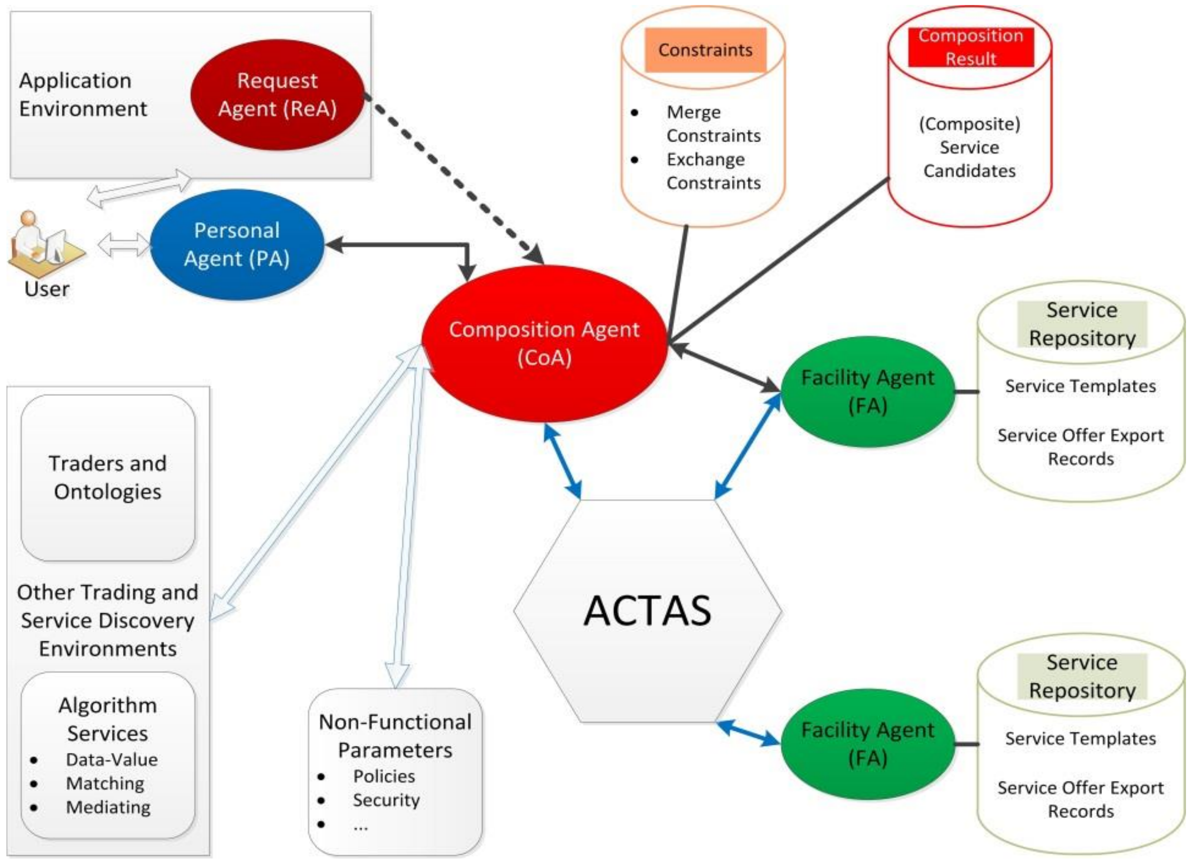


Fig. 23 - Phase 4

During the checking of Merge Constraints, the so-called merging of the Service Properties of principally compatible Service Ports takes place. Every Service Property was as a Char Property declared in the same Compatibility Characteristic. Thus, pairs of comparable Service Properties are created through the principal compatibility holding the same Merge Property Classes. The Merge Property Class is a mean for the import and realisation of matching algorithms and if necessary mediation algorithms, in order to cope with the case that the data of the merging Service Properties are based on different ontologies or units. The Merge Property Classes are used for the creation of Merge Property Objects

Other constraints could be checked directly by the Facility Agents. The Facility Agents could offer a resource management, which could include a reservation of resources for services, in order to ensure that the service could be deployed later on. This demands a transaction control and locking.

8.5 Phase 5 – Grounding and Schedule of the Services

In the fifth phase of the Service life cycle, a found Service Candidate has to be grounded, i.e. a Concrete Service has to be found, which can be deployed and executed. Therefore, the Service Grounding can involve the recursive call of other Trading and Service Discovery environments.

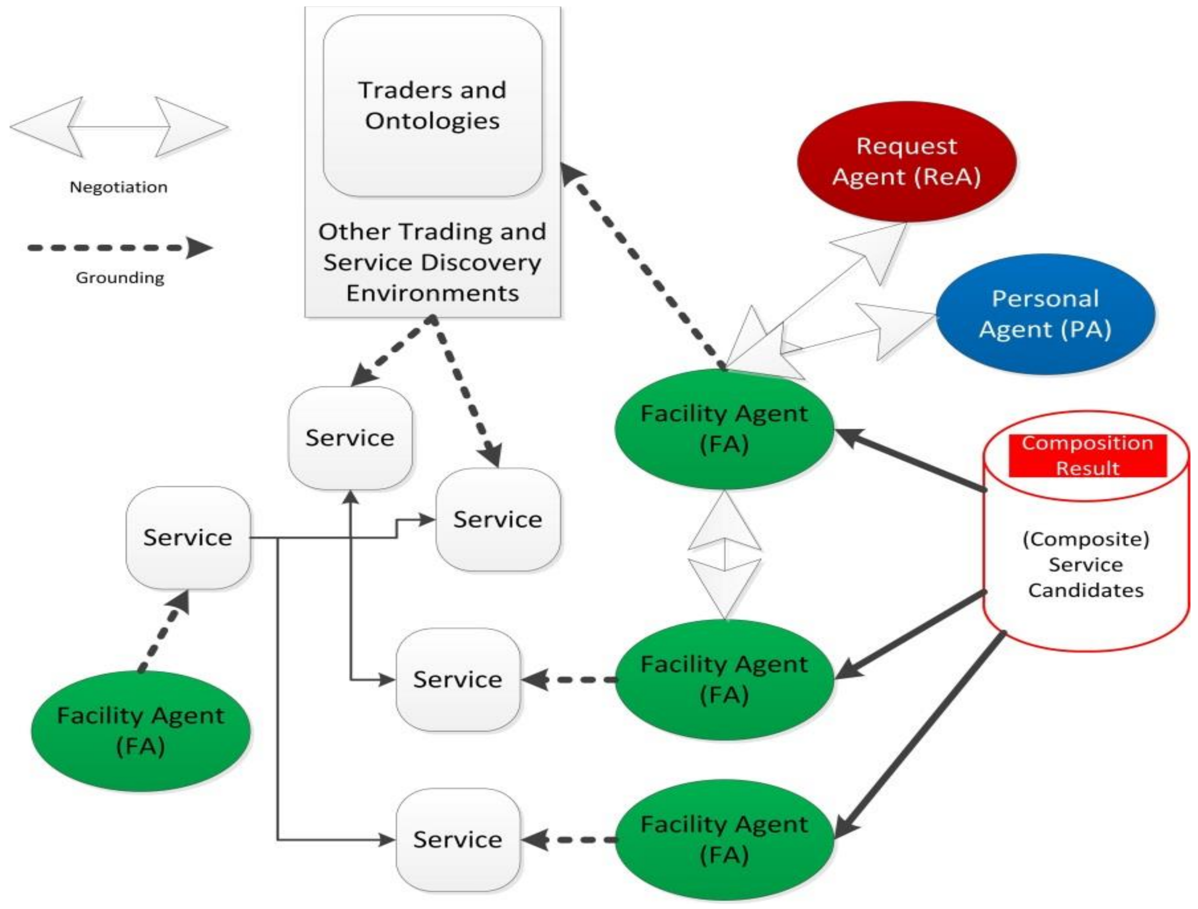


Fig. 24 - Phase 5

However, before a Service Grounding can take place, possibly, a Service Candidate has to be selected from several ones found as Composition Result in phase 4. Facility Agent will always check the Service Offer Export Records of the Service Candidates, in order to find out if the used Service Offers are still valid.

The information for the Service Grounding might be at least partly already known to the Facility Agent. Nevertheless, the settlement of the Service Candidate and the getting of the Service Grounding information demand a negotiation with the other involved Facility Agents (Fig. 24). The goal of the negotiation is to find a Service Level Agreement (SLA) for the most prospective Service Candidate. The Request Agent, the Personal Agents as well as the potential Service Clients and Service Providers can also be involved in the negotiation. It is an advantage, that ACTAS already includes agents for an active negotiation and coordination of services. The Service Requester might be involved in confirming the results of the Service Composition with the application environment.

8.6 Phase 6 – Service Execution and Feedback

In a last phase, the Service Deployment and Service Execution/Consumption take place. There will take place a direct communication between the services and the Service Clients in the end (Fig. 25). Additionally, the feedback of the Service Clients can be of interest.

The Service Composition includes the Service Clients as so-called **actors**. In the data-structure of the Composite Service, the actors are represented through so-called **Actor Service Offer**s (**ASO**). In future research, ACTAS will use the client feedback for learning. A so-called Actor

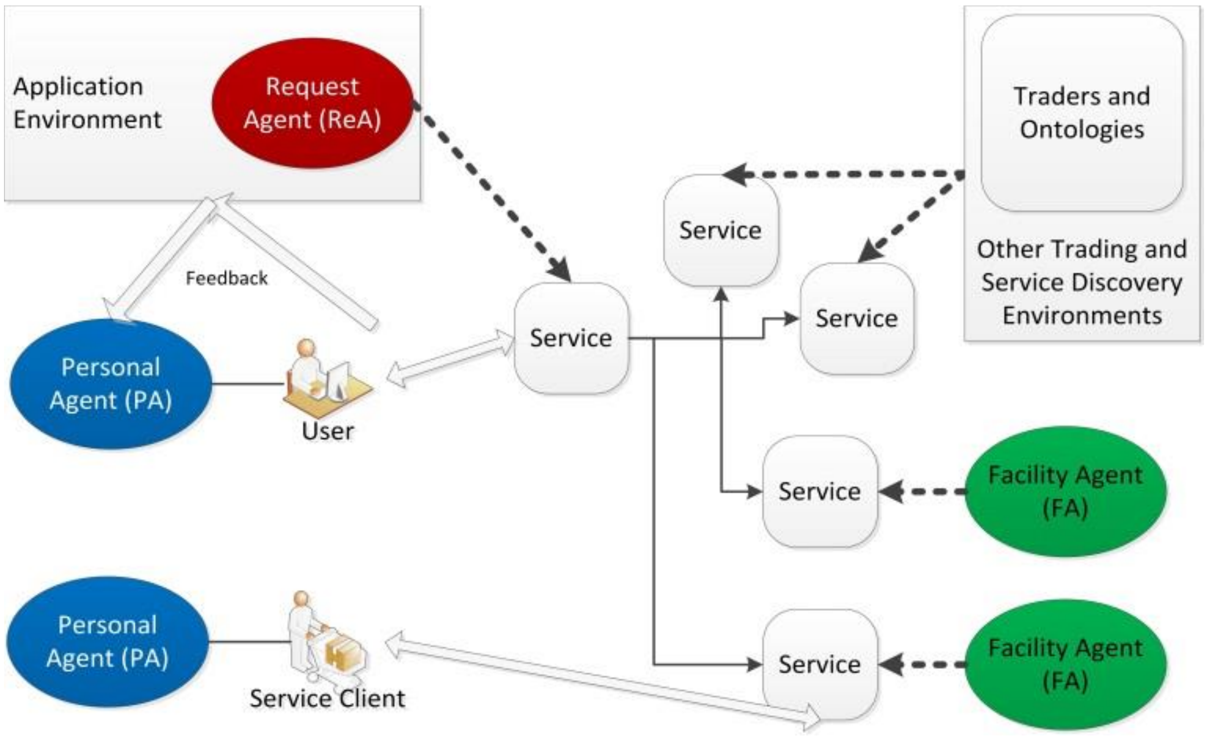


Fig. 25 - Phase 6

Service Template (AST), stored in the Personal Agent of a Service Client, will collect for every Service Request the Facility Agents of services recognized of high quality (QoS). Afterwards, the AST could be used for the initialisation of the ASO of the Service Client for an identical Client Service Request, in order to address the collected FAs prioritised.

It will be the concern of future research to consider further ways how a so-called **Actor Service Template** (**AST**) can help in the generation of an ASO. Another goal of future research will be the support of the monitoring of service qualities during the Service Execution. When a Component Service had to be replace due to bad quality issues, then ACTAS could be ask for delivering of a replacement Component Service.

ACTAS - SERVICE MODEL (S-MODEL)

9 Introduction to Service Model (S-Model)

ACTAS distinguishes between S-Model, R-Model, and C-Model. The support of the Service Design (Phase 1 of the extended life cycle of servicea as described in section 2.3.4 - 4th aspect of services: The extended life cycle of services) is covered with the S-Model. The R-Model is an extension of the S-Model and covers the information data structures of the Service Request and the Trading Request. The active phases of Service Discovery (Service Trading (Phase 2), Service Matching (Phase 3), and Service Checking/Selection (Phase 4), cf. section 2.3.4 for life cycle as a general aspect of services) are addressed with the C-Model, introduced in chapter 12. This chapter introduces the building elements of the S-Model, whereas the next chapter presents the Service Description of ACTAS more in detail.

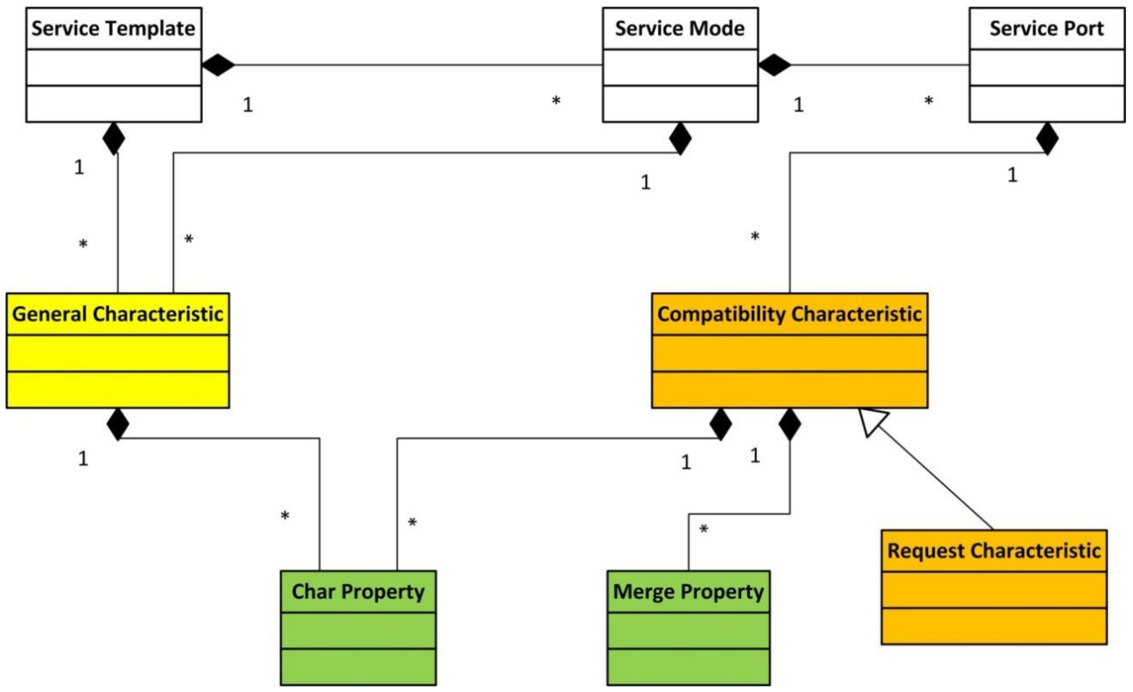


Fig. 26 - Elements of the S-Model

In order to design a more general Service Description, it was essential to get clarified what a service is. This was the research interest of many scientists in the past as shown in the State-of-the-Art. Additionally, it turned out that the aspired automated discovery, composition, and execution of services through the agents need a support for reasoning about services. Some approaches like METEOR-S (cf. section 5.2.1) work with separated repositories for Service Offers of different domains. However, section 2.3 (Aspects of service) showed that services could be categorized by other criteria than the point of view of their application domain.

Alternatively, ACTAS proposes a multi-ontological categorization of the area of services. The services are not any longer kept in separated repositories, but dynamically classified through the association of commonly agreed Semantic Characteristics, which can be recognized by the agents.

Generally, an ontology categorizes the instances of the problem domain through its schema, which consists of concepts, relations, and attributes. A repository of instances together with their ontology schemas is called an **ontological repository** in this thesis. It is essential to know what kind of instances is addressed through the named concepts of an ontology schema, in order to choose the fitting constructors for expressions over concepts of different ontological repositories. In the case of the same kind of instances, the concepts can be directly combined (i.e. through intersection or union); otherwise relations have to be defined between the instances. Table 5 - Constructors of DL showed constructors with classifying concepts for the application of these instance relations inside of logical expressions.

Services as instances can be categorized based on various criteria like e.g. their application domains, the user roles of their Service Clients, statements about their life-cycle phases as for example the degree of abstraction of their Service Description, or the supported non-functional properties in the different phases. The later example combines the 2nd aspect (non-functional attributes) with the 4th aspect (phases of the life cycle) (cf. section 2.3). Ontologies, which provide schemas for these criteria, exist. In the appendix A, for instance nfp-ontologies are listed.

ACTAS introduces so-called Semantic Characteristics as commonly agreed concepts for the categorization of services, which are defined through the combination of concepts of “criteria ontologies”. Each Semantic Characteristic can become a free composition of ontological classified criteria, i.e. it gains a semantic meaning. For example, a concept of an ontology for Service Design classifying services according to certain Service Description formalisms (e.g. WSMO, OWL-S, or WSDL) could be intersected with a concept of another ontology categorizing services according their application domain (e.g. communication services, Customer Relationship Management (CRM), processing of geographic data) (cf. section 2.3.1). A result could be a characteristic defined for services as instances doing Customer Relationship Management (CRM), which are offered through a WSMO Service Description. It is obvious, that the WSMO Service Descriptions of this service category will be comparable since they are based on similar domain ontologies.

A Service Offer in ACTAS shows its categorization through its associated set of Semantic Characteristics. A Service Request can just ask for a service of this category through demanding the same set of Semantic Characteristics. Following this model, a characteristic can as a first purpose support the Service Discovery and Service Selection as they describe the principal compatibility of services. Semantic Characteristics for this purpose are called **Compatibility Characteristic**s. Two services are called **principally compatible** when they can agree on one Service Mode holding the same set of Compatibility Characteristics. Therefore, a Compatibility Characteristic can be seen as a commonly agreed, standardized delimitation of the area of services, in order to support the Service Discovery, Service Matching, and Service Selection. A special kind of Compatibility Characteristic useable in a Service Request is a **Request Characteristic**. A Request Characteristic is related with an ontology categorizing user roles. Therefore, it possibly indicates a restriction of the user group, but also allows an adapted Service Description for this user group.

Secondly, a Semantic Characteristic is a commonly agreed semantic context for properties, similar the attributes of an ontology. These properties are called Char Properties. However in contrast to the attributes of ontologies, algorithms are linked with a **Char Property**, in order to control its content. Therefore, a Semantic Characteristic is also an agreement on algorithms. This leads to mentioned support of Compatibility Characteristics for the Service Matching (Phase 3 of the life cycle, cf. Fig. 2). Since principal compatibility means identical sets of Compatibility Characteristics, it also causes the same sets of Char Properties. In the terms of ACTAS, the properties of two services become "comparable" and their "merge" algorithms can be used for the checking of matching constraints involving possibly mediation on the level of each Char Property. The application of mediation might become necessary, when for instance the data of comparable properties are based on different ontologies. In ACTAS, this is called the merge of comparable properties and the checking of **Merge Constraint**s.

Thirdly, a Semantic Characteristic opens up the possibility to access annotating criteria in a commonly agreed way. In the discussion about non-functional properties (section 2.3.2) (nfp) of services, it was mentioned that some of them have annotating purpose (e.g. the information about the provider, cf. appendix section A). Normally, this kind of nfp is not involved in the Service Discovery, but they are possibly useful for the Service Selection (Phase 4) and Service Negotiation (Phase 5). Thus, it is of advantage to have commonly agreed access methods of such properties. Therefore, a so-called **General Characteristic** can be introduced in ACTAS, which is related to a concept of an ontology classifying such an annotating criterion. Associated with a service, such a General Characteristic indicates the support of this annotation to the agents.

ACTAS does not work with explicit service repositories. The facility agents (FA) have the active role of publishing Service Offers (SO) through a Service Offer Export Record (SOER) based on Service Templates (ST). A ST consists of several Service Modes, which are defined through their distinct set of Semantic Characteristics. A FA should be proactively able to decide whether it can offer a service candidate if necessary through an additional Service Mode, when a Service Request asks for a specific set of Service Characteristics, which the FA could provide. For this purpose, the FA has to know the semantic relationships of Service Characteristics. The ontology of the Service Characteristic should support this decision of the FA. The publication through ST and SOER allows the FA firstly to keep control over the Service Modes of a service, secondly it can adapt the Service Offer to the current state of the service, and the FA could thirdly "reserve" resources for the deployment of a service in a selected Service Mode.

The Service Description in ACTAS through Service Templates and Service Offers, respectively, leads to a fourth purpose of Sematic Characteristics: acting as "building blocks" of the Service Descriptions. Compatibility Characteristics and their special kind, the Request

Characteristics, are used for the description of compatibility. Annotating criteria are added with General Characteristics. The Char Properties are linked with algorithms for the access and merge, in order to support constraints of their values (Attribute Constraints, Value Constraints) and constraints of their so-called merge (Merge Constraints). Char Properties appearing in a Service Description are called Service Properties. Finally, Exchange Constraints declare constraints between different Service Properties. In this context, General Characteristics can be introduced, which offer Char Properties for the holding of accumulated results of Exchange Constraints. For instance reliability (an nfp, which is not only annotating, cf. section 2.3.2) could be checked for each Component Service of a Composite Service. In the case that reliability is also of interest on the level of the Composite Service, a General Characteristic of the mentioned type could offer the fitting algorithms with its Char Properties. Together with Exchange Constraints involving the reliability values of the Component Services and the Composite Service, reliable Composite Service could be selected in phase 4.

Inside of the declarative environment of ACTAS covered with the C-Model, the Char Properties and necessary entities for the access of the Char Properties inside of the constraints appear as objects, which wrap algorithms/methods. This leads to the framework character of ACTAS. The linked methods allow an initialisation, setting, and checking of the Char Properties and its constraints. In Definition 6, Property Classes are introduced for the description of these "objects". Ontological repositories of **Property Class**es, Semantic Characteristics, and eventually Exchange Constraints offer the "building blocks" to the Service Designer of ACTAS.

Element	Description
Semantic Characteristic (Char) • General Characteristic (GCh) • Compatibility Characteristic (CCh) • Request Characteristic (RCh)	• A (Semantic) Characteristic is an ontological concept for the categorization of services through several aspects. (Service Ontology with Characteristics as concepts.) • It can be used for the description of compatibility between services (CCh). • Service Requests can be given for specified user groups (RCh). • It is the semantic context for the declaration of a set of Char Properties. • Integration of (established) algorithms linked with Char Properties and constraints. • The semantic context also allows the initialization of the Char Properties with specific Value Constraints including fitting Application Ontologies through the linked algorithms. • Standardized access of annotating non-functional properties (GCh). • Building blocks for the Service Design, in order to create Service Description together with Exchange Constraints (GCh and CCh). This leads to ontological Repositories for the Semantic Characteristics.

Element	Description
Property Class • Char Property Class • Merge Property Class • Exchange Property Class • Service Property	• In the description of a Semantic Characteristic and of an Exchange Constraint, the Property Class is a pointer to an algorithm. This can be for instance a Web Service or a Prolog module. (The latter extending the declarative environment of ACTAS directly.) • In the declarative environment of ACTAS, the Service Properties appear as objects, which wrap as adaptors the access of the algorithms. • Any Char Property whose Characteristic is part of a Service Description is called a **Service Property**. • A Char Property offers methods for the setting of constraints for the values of its attributes (Value Constraints). Other methods will allow the return of information about the current attribute values. • A Merge Property is declared for the "merge" of two Char Properties of the same Property Class. Its merge method can wrap established matching and mediation algorithms. • An Exchange Property offers a set of methods for the exchange of several Char Properties of different classes.
Constraints • Attribute Constraints • Value Constraints (Va-Co) • Merge Constraints (Me-Co) • Exchange Constraints (Ex-Co)	• The algorithms of the Property Classes, especially the ones of Char Properties, include naturally constraints for the values of its attributes. These so-called Attribute Constraints cannot be changed. • The semantic environment of a Characteristic, a Service Template, and a Service Offer (through the SOER) describe additional Value Constraints for the Char/Service Properties. The Value Constraints in a SOER can overwrite the ones in a ST. The Value Constraints use the methods of the Char Property Classes. • With each Char Property in a Compatibility Characteristic is linked a Merge Property Class. In the C-Model, the merge algorithms of this class are used for a compatibility check. These so-called Merge Constraints combine two Char Properties of the same class. They allow the framework like inclusion of established matching and mediation algorithms. Exchange Constraints are built over several Service Properties declared with different Property Classes. They use methods of Exchange Properties. Exchange Constraints allow the checking of constraints between Service Properties appearing in different contexts of a Composite Service (cf. C-Model).

Element	Description
Service Template (ST) • Service Mode (SM) • Service Port (SP)	• A ST is a Service Description of a service, which sums up different Service Modes how a FA can principally offer a service. • The Service Modes have Service Ports, in order to describe the compatibility of services with the use of Compatibility Characteristics. • The Facility Agents (FA) in the model of ACTAS exports Service Templates as XML-files. • Service Templates could be used by Trader Agents for a principal composition of new Composite Services, which could be offered by the Trader Agent as a FA. • A FA can dynamically extend a ST with another Service Mode, when it recognizes that it could deliver the Compatibility Characteristics requested by a Service Request (cf. C-Model).
Service Offer Export Record (SOER) • Service Offer (SO)	• ACTAS supports the availability control of services through the distinction between Service Offer and Service Template. Therefore a FA exports SOERs related to a certain ST as XML-files. • Through a SOER, the FA determines the validity of a SO based on a ST. A SOER contains a time stamp and a unique identity. • A SOER describes the currently available Service Modes and can overwrite the Value Constraints of the Service Template. • A SO is built from the ST and a current SOER in the declarative environment of an agent. • A FA can offer the possibility to reserve a selected Service Mode of a Component Service in a found Composite Service (cf. C-Model) on the basis of a specific SOER. • ACTAS observes the availability of services. Therefore, a FA exports a description of an available service with a SOER, which is based on a Service Template. The SOER is also an XML-File. It describes the currently available Service Modes and the values of the properties. The latter are set by so-called Value Constraints.

Table 12 - Elements of the S-Model

9.1 S-Model: Semantic Characteristics

A Semantic Characteristic is a commonly agreed ontological concept for the categorization of services, and additionally a semantic context for the description and declaration of properties (especially so-called Char Properties) through Property Classes (introduced in the next section). The constraints on values of a property given with its Property Classes (so-called **Attribute Constraint**s) can be further specified through **Value Constraint**s (Va-Co) (Constraints are discussed in section 10.4), in order to adapt the values to the semantic context. In the models of

ACTAS (S-Model and C-Model), Characteristics are a key-element and are used for two purposes: Firstly, the description of services in general and secondly the description of their compatibility. Together with **Exchange Constraint**s, which describe constraints involving several Char Properties, they become like "building blocks" for Service Descriptions and Service Requests.

Definition 4. (Semantic) Characteristic (Char)

$$\mathrm{GCh} \equiv (\mathrm{GCh}^{\mathrm{ID}}, \mathrm{SemDescr}, \mathrm{GCh\_Property}^{\mathrm{Set}}, \mathrm{Char\_Env})$$
$$\mathrm{CCh} \equiv (\mathrm{CCh}^{\mathrm{ID}}, \mathrm{SemDescr}, \mathrm{CCh\_Property}^{\mathrm{Set}}, \mathrm{Char\_Env})$$

$$\mathrm{GCh\_Property} \equiv (\mathrm{Name},\ \mathrm{CharProperty}^{\mathrm{Class}})$$
$$\mathrm{CCh\_Property} \equiv (\mathrm{Name},\ \mathrm{CharProperty}^{\mathrm{Class}},\ \mathrm{MergeProperty}^{\mathrm{Class}})$$

$$\mathrm{Char\_Env} \equiv (\mathrm{Char\_Va\_Co}^{\mathrm{Set}}, \mathrm{Char\_Ex\_Co}^{\mathrm{Set}})$$

There are two different kinds of characteristics:

- **General Characteristic** (**GCh**)
- **Compatibility Characteristic** (**CCh**)

They are described by:

- Name or identification of characteristic ($\mathrm{GCh}^{\mathrm{ID}}$, $\mathrm{CCh}^{\mathrm{ID}}$)
- Description of the semantic/ontology of the Characteristic (SemDescr) (cf. Definition 5)
- Their environment (Char_Env), which contains the settings of the value-constraints of the Property Classes (($\mathrm{Char\_Va\_Co}^{\mathrm{Set}}$) and the exchange-constraints ($\mathrm{Char\_Ex\_Co}^{\mathrm{Set}}$).
- The Properties of the characteristics are described by Property Classes (cf. Definition 6)
- A **Request Characteristic** (**RCh**) is a Compatibility Characteristic related to user groups in the semantic description

In ACTAS, the Service Providers describe the services. Therefore, the description is done mainly from the perspective of a Service Provider. **Compatibility Characteristic**s are used for the description of compatibility between services as well as between a service and a request. A service is called principally compatible with another one if it holds the same Compatibility Characteristic(s). A special kind of Compatibility Characteristic is a **Request Characteristic** used for the description of Service Requests and their compatibility to Service Offers. The definition of the ontological concept linked with a Request Characteristic includes the relation

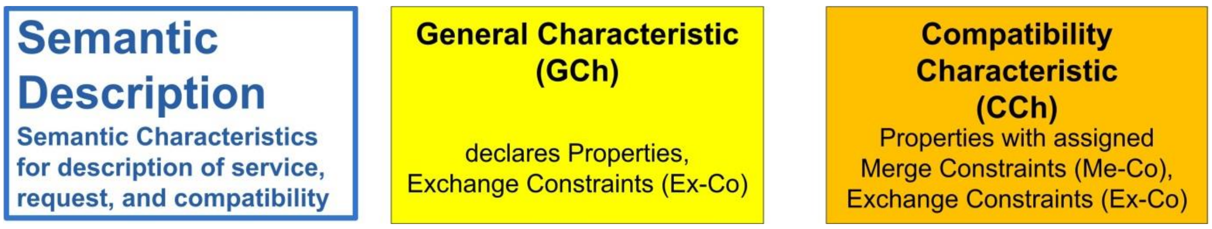

Semantic Description
Semantic Characteristics for description of service, request, and compatibility
General Characteristic (GCh)
declares Properties, Exchange Constraints (Ex-Co)
Compatibility Characteristic (CCh)
Properties with assigned Merge Constraints (Me-Co), Exchange Constraints (Ex-Co)

Fig. 27 - Semantic Description

between services and user groups, i.e. a Request Characteristic restricts the semantic context of Service Requests and of their Char Properties to specific user groups. **General Characteristic**s allow a standardized access of annotating properties and the introduction of new Char Properties, in order to hold values resulting of Exchange Constraints.

In Definition 4, General and Compatibility Characteristics are introduced. The obvious distinction between both kinds of Semantic Characteristics is their declaration of a property (also called **Char Property**): a property of a Compatibility Characteristic (CCh_Property) has got an additional association of a Merge Property Class ($\text{MergeProperty}^{\text{Class}}$). The Merge Property Class is used for the Merge Constraints (cf. section 10.4) checking the matching of two services on the level of their Char Properties.

Definition 5. Semantic Description (SemDescr) of Char

$\text{SemDescr} \equiv (\text{Is-a}^{\text{Set}}, \text{logOntologyRelation}^{Expr})$

- The semantic description SemDescr of a characteristic Char has an "Is-a" set of inherited characteristics (multi-inheritance). The result is an ontological schema of Semantic Characteristics.
- The set of entities classified by Char is a subset of each set of entities classified by a characteristic being element of the "Is-a" set (corresponds to the "$\sqsubseteq$" relation in DL (cf. section 4.2).
- The set of entities classified by Char is additionally restricted through the interpretation of the logical expression $\text{logOntologyRelation}^{Expr}$. In the case of DL, the expression is built und semantically interpreted with the constructors in Table 5.
- The domain set Δ^I of the characteristic ontology, i.e. the set of all objects in the ABox in DL, is the set of all services/Service Offers. The semantic interpretation I of a Char with semantic description
  - $\text{SemDescr} = (\{F_1, \cdots, F_n\}, E)$ is:
  $$Char^I \equiv \{s \in \Delta^I | s \in (F_1{}^I \cap \cdots \cap F_n{}^I \cap E^I)\}$$
- Properties and environments (cf. Definition 4) of Semantic Characteristics are not inherited

The environment of a characteristic (Char_Env) is built from the Value Constraints and Exchange Constraints, in order to fix the constraints, which are relevant in its semantic context. The Value Constraints define restrictions for the Char Properties, which are generally valid in the semantic context of the characteristic. The Exchange Constraints define constraints involving several Char Properties, when certain premises are fulfilled. The constraints in a Semantic Characteristic cannot be changed by the Service Description. They are the constraints, which are checked in the algorithms of the C-Model with the highest priority. Further information about constraints is in section 10.4 - Constraints in the Service Description.

The semantic of the Semantic Characteristics is defined through a logic expression over concepts and a „is-a"-relationship inside the ontological schema of the characteristic as concepts (cf. Definition 5). The instances of this characteristic ontology, i.e. the elements of the ABox in a

DL based ontology, are **services**. This is due to the fact that Semantic Characteristics are first of all a commonly agreed classification of concepts over services. In DL, the "is-a"-relationship is shown with the operator/constructor "$\sqsubseteq$" (cf. section 4.2). This relation besides others is illustrated in Fig. 28. The top concept of "Semantic Characteristic" is directly distinguished into the concepts of General Characteristics (GCh) and Compatibility Characteristics (CCh).

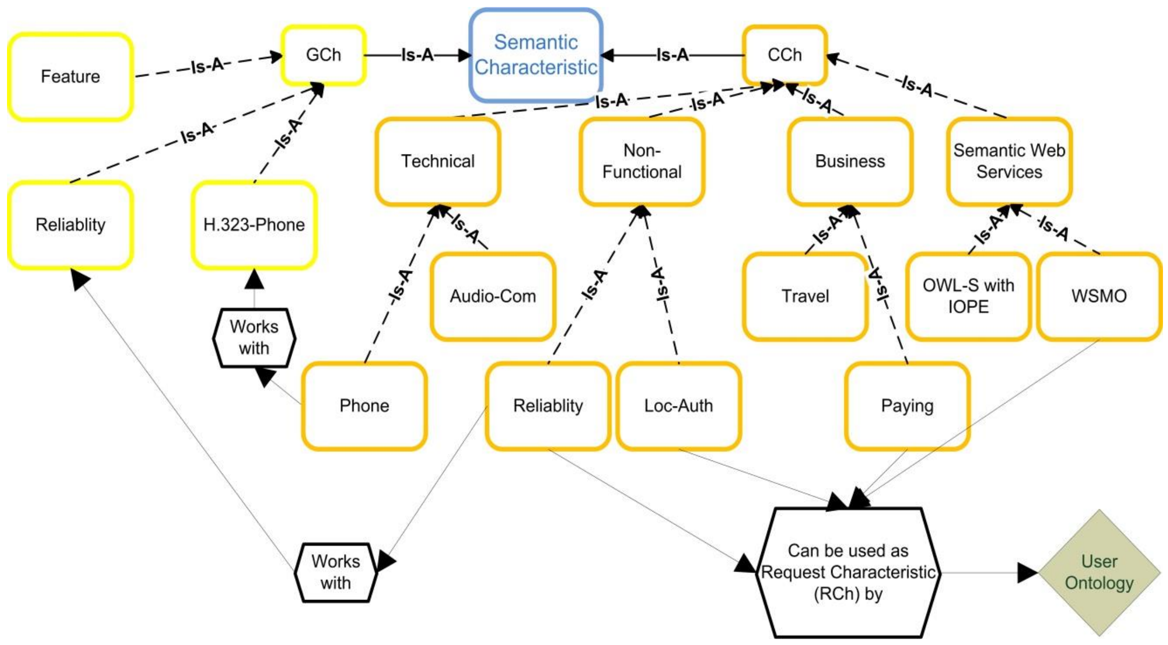


Fig. 28 - Principal ontological categorization of Semantic Characteristics

The logic expression for the definition of the semantic of characteristics contains terms with negation, disjunction, and intersection (conjunction) of concepts, when these concepts also classify services. The constructors of DL for these logical expressions are given in Table 5. When the instances of concepts are not directly services then the relational constructors are used. The user ontology mentioned in Fig. 28, which classifies user groups/roles as instances, is an example in this direction. Request Characteristics are defined through relations to certain user groups. Further additional ontologies can be introduced, which offer concept schemas for the various aspects of services (section 2.3).

$$\begin{aligned}\mathrm{GeodataWSMOTrader} \coloneqq \mathrm{SemDescr}(\{\mathrm{SWS}-\mathrm{CCh}\}, [(&\mathrm{Ontology}_{\mathrm{Design}}, \mathrm{WSMOgeneral}) \sqcap \\ &(\mathrm{Ontology}_{\mathrm{Domain}}, \mathrm{Geodata}) \sqcap \\ &(\mathrm{Ontology}_{\mathrm{Trader}}, \mathrm{GeodataEnvA})]) \end{aligned} \tag{9-1}$$

The "Is-a" relation means an inheritance of semantic descriptions. In order not to restrict the declaration of Char Properties, ACTAS defines the "Is-a" inheritance (cf. Definition 5) only on the semantic description, i.e. the logical expression of a semantic descriptions is conjugated with the inherited Semantic Description (cf. Definition 5). The Char Properties can be freely defined, but should certainly cover the semantic. In the evaluation (chapter 14), simple semantic descriptions of characteristics are discussed. Afterwards, the extended usability of characteristics is considered.

In equation (9-1), an example of a semantic description of a Compatibility Characteristic is shown that combines criteria concepts for services with SWS Service Description through WSMO originated in an imagined Geodata domain with a concept for trading (4th aspect – phase 2). A concept in an ontology can also have properties, as the WSML ontologies listed in the chapter B of the appendix show. The latter concept in the semantic description is assumed to mention a Service Discovery environment A in the Geodata domain in one of its properties. It is likely that this environment includes traders of its own. In the subsequent example, an idea is developed, which takes advantage of Semantic Characteristics, which are semantically connected with a trading criteria that even contains a link to an external trading environment.

Example 5 **Addressing of Trading Phase through Semantic Characteristic**

A Compatibility Characteristic can be connected in its Semantic Description with the Trading Phase in such a way, that certain Trader Agents (TrA) of the ACTAS System Environment will react on a Service/Trading Request containing this Compatibility Characteristic. The Trader Agent might directly access a Trading Environment of an existing Service Discovery Environment, in order to find a compatible service.

However, the concrete environment is normally unknown to the Service Designer. The main idea is the active involvement of the Trader Agent (TrA) as described in Example 5. The pro-active and re-active behaviour of the Trader Agents allows the creation of Trader Agents, which react on Compatibility Characteristics holding this specific trading concept/criterion. In this way, ACTAS supports an adaptive trading behaviour controlled through Semantic Characteristics, which can involve existing trading environments. The Service Designer can control the trading by using Semantic Characteristic associated with such a criterion in his Service Description/Service Request.

Example 6 **A Semantic Characteristic for the Deployment Phase**

Due to the fast development of Web Service standards, it turned out that it was important for the deployment that the Service Requester and the Service Provider agreed on a certain set of Web Service standards and their versions. In the state of the art (section 3.7), it was mentioned for example that the high number of Web Service standards and their incompatibility of versions was addressed through **WS-Interoperability** (**WS-I**) [OAS2011]. A Compatibility Characteristic could verify the use of methods in this direction.

Since Semantic Characteristics are concepts over service instances, the Service Designer can extend his/her control of early Service Selection to criteria coming up in later phases of the life cycle (4th aspect of services). Through the use of semantically fitting characteristics, it can be ensured that the (component) services can be deployed and executed in such a way that they can work together. In Example 6, a Semantic Characteristic, i.e. a Compatibility Characteristic, is discussed, which could ensure that compatible packets of Web Service Standards are used. Other potential Compatibility Characteristics allow an early agreement on certain methods of choreography and orchestration, which are criteria of the phase 5 of the life cycle of services (cf. section 2.3.4). Extended, more abstract settlements in this phase of the life cycle, as for instance the negotiation about the company policy of the enclosement of orchestration details, led to criteria belonging to the 3rd aspect of services, the inherent complexity of a service (cf. section 2.3.3). In the next section, the Property Classes are discussed, which are necessary for the declaration of the properties of Semantic Characteristics. They allow the taking over of established algorithms of Service Discovery and Service Matching as well as the construction of constraints.

9.2 S-Model: Property Classes

Definition 6. Property Class (PC)

$\text{PC} \equiv (\text{PC}^{\text{ID}}, \text{InstanceOf}^{\text{Set}}, \text{GroundingDescr}, \text{Method}^{\text{Set}})$

PC is an object oriented class definition:

- PC^{ID} is a unique identification for the Property Class,
- $\text{InstanceOf}^{\text{Set}}$ categorizes the Property Class in an ontology of property concepts (cf. 4.2) (A-Box in DL)
- $Method^{Set}$ enumeration of the methods implemented by the class
- GroundingDescr is information about the access of the algorithm, i.e. the way how the access has to be wrapped by the environment of ACTAS as well as the way how to realise the algorithm for instance as a (statefull) Web Service having a WSDL

Fitting with their semantic context, Semantic Characteristics wrap Char Properties as well as an environment (Char_Env), which describes **Value Constraint**s ($\text{Char\_Va\_Co}^{\text{Set}}$) and **Exchange Constraint**s ($\text{Char\_Ex\_Co}^{\text{Set}}$) relevant in this semantic context (cf. Definition 4). The Char Properties declared in the semantic context of Compatibility Characteristics are also used for the compatibility description (cf. section 10.4.2). In all this cases, so-called Property Classes (PC, Definition 6) are involved. ACTAS introduces three different kinds of Property Classes (cf. Definition 7, Fig. 29). They create a framework like adaptivity of ACTAS: (1) The Char Properties are declared with Char Property Classes. (2) Value Constraints use methods of the Char Property Classes, in order to set (restrictions for) the values of the Char Properties. (3) The Char Properties in Compatibility Characteristics are additionally associated with Merge Property Classes for the import of matching and mediation algorithms, which are used for the so-called **Merge Constraint**s. (4) Finally, Exchange Property Classes are used for the import of algorithms for the definition of Exchange Constraints, in order to correlate several Char Properties.

The S-Model proposes an ontological repository of Property Classes, in order to achieve a "standardisation" of Service Properties and their methods as well as to support the use of characteristics as "building blocks" of Service Description. The concepts of the ontology of this ontological repository could classify with the help of the different application domains and functional aspects. The $\text{InstanceOf}^{\text{Set}}$ information in the Definition 6 of a Property Class (PC) can

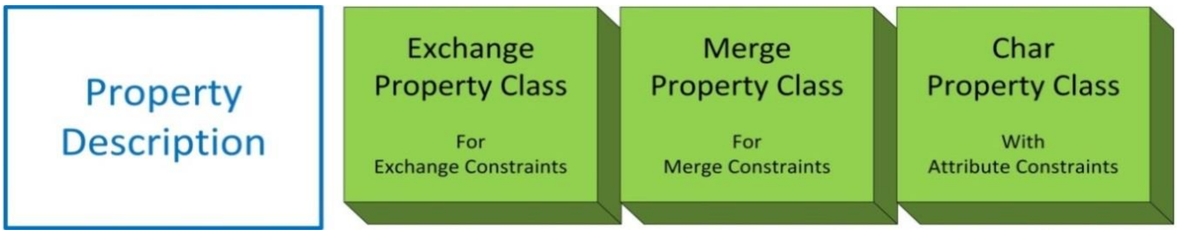


Fig. 29 - Property Classes

be seen as a categorization of the PC instance with a set of classifying concepts in ontological repositories.

A Property Class includes a behavioural semantic, i.e. it provides methods for value setting and managing of constraints as well as testing. Each of these methods reflects the success of its operation back to the declarative environment of ACTAS through a Boolean functional result. This behavioural semantic allows the integration and probing of existing, established algorithms of Service Discovery and Service Matching including mediation.

Example 7 **Principal Char Property Class**

> Let the dealing with Capability Descriptions of OWL-S be implemented in a Char Property Class with the $\mathsf{PC}^{\mathsf{ID}}$=OWL-S-capability. Then this Property Class can be used for the declaration of a Char Property in the Compatibility Characteristic “OWL-S_with_IOPE” introduced in equation (22-18) of Table 34 - Request Characteristics (RCh) in the appendix. A method will initialize the Char Property with a capability description of an OWL-S description. This method will have a parameter which allows the access of an existing OWL-S description through a URL. It is likely that the Property Class will include a plausibility check of the IOPE-parameters as Attribute Constraints.

ACTAS introduces three different kinds of Property Classes used for the declaration of Char Properties, Merge Properties, and Exchange Properties (cf. Definition 7). A set of methods ($\mathsf{Method}^{\mathsf{Set}}$) is listed in the general Property Class (PC) defined in Definition 6. This $\mathsf{Method}^{\mathsf{Set}}$, an enumeration of method specifications, can be understood as an application interface (API), being of interest for the ACTAS and Service Administrator/Designer (cf. chapter 8 - System Environment). The methods of a PC enable the setting of values like e.g. the initialisation of the Char Property mentioned in Example 7. The Service Designer will use this method interfaces for the description of the Value Constraints and Exchange Constraints in the Service Description. The constraints in the environment of a Semantic Characteristic (Char_Env in Definition 4) are written with the method specifications by an ACTAS Administrator.

A Char Property Class should offer methods for the setting of Value Constraint, the testing of values, and a method “printValues” for the producing of information about its current values. An Exchange Property class must have methods for the checking of the values of several properties, simultaneously. The $\mathsf{Method}^{\mathsf{Set}}$ of the Merge Constraints is determined through the needs of the Composition Process in the C-Model, since the Merge Constraints are an essential part of testing the compatibility (cf. section 10.4.2). Therefore the methods of Merge Constraints are discussed in the context of the C-Model. The C-Model will translate the methods of the Property Classes into declarative operation, i.e. predicates.

However, since ACTAS acts as a framework, it cannot control that every use of the methods leads to unambiguous attribute values in the objects. This ambiguity might be even wished. Therefore, ACTAS has to speak generally of constraints leaving the details up to the encapsulated environments of the Property Classes. However, ACTAS demands a monotony of the constraints. The successful application of a constraint must be reflected in a Boolean return value. In the C-Model, it is motivated and discussed, how the Composition Process could query for alternative applications of a constraint. In the S-Model, the constraints are discussed more in detail in section 10.4.

Definition 7. Kinds of Property Classes

Three different kinds of Property Classes are distinguished:

- A **Char Property** is declared through a **Char Property Class** in the semantic context of a Semantic Characteristic (CCh_Property and GCh_Property in Definition 4). Its methods have only parameters and.
- A **Merge Property Class** is linked with a Char Property of a Compatibility Characteristic in the CCh_Property (cf. Definition 4). Its "merge" method processes two Char Properties declared with the same Char Property Class, in order to test their matching (potentially including mediation if their values are based on different domain/application ontologies). This so-called Merge Constraint can lead to a Merge Property that can be tested for further criteria.
- **Exchange Property Class**es are used in an Exchange Constraint, i.e. a constraint correlating several Char Properties possibly declared with different Char Property Classes. Their methods can process Char Properties of different Char Property Classes, in order to see if their values can be adapted for the fulfilment of constraints. This can be interpreted as a translation/mediation of the values of some Char Properties into the values of some other Char Properties.

ACTAS uses objects in the declarative environment for the wrapping of the access of the implementations of the classes.

In the declarative environment of ACTAS, the access of the properties is comparable to objects[5] (in fact, the object extension of SICStus Prolog was used; cf. Evaluation, chapter 14). However, the declarative environment of ACTAS only wraps the access of the implementation of the properties. The implementation of the methods of a Property Class can be externally. The description of grounding (*GroundingDescr* in Definition 6) contains more information. For instance similar to the WSML description of a mediator in WSMO (cf. [StGrAb2007 section(s) 287–311]), which addresses a Web Service implementing the mediation; it could contain the URL of a Web Service, i.e. of a WSDL description. Every new "object", created for such a

[5] In the thesis the terms "Property Class" and "Property Object" are used.

Property Class in the declarative environment, should access the Web Service in its own context. It definitely helps, when the Web Service is statefull for each of these contexts, in order to keep hold of the (internal) constraints. Otherwise, ACTAS must manage the settings and will have to send complete lists of settings each time, when it accesses the Web Service, in order to test their fulfilment. This keeping track of the already applied constraints creates a monotony of the constraints and it is necessary for the integration in the declarative environment.

The challenges of the integration of objects in the declarative environment of ACTAS are discussed in the C-Model (chapter 13). The next chapter introduces the Service Design of ACTAS. In this context, specific methods, which should be offered by the Property Classes, are discussed. The Service Design description ends with a closer look at the constraints. Since constraints are based on method calls as sketched in this chapter, the control of the method calls will be further considered in section 10.4.

10 Service Description

The Service Model (S-Model) distinguishes between Service Templates (ST) and Service Offer Export Records (SOER) for the Service Description. The Facility Agents, which are under control of the Service Providers, are responsible for the publication and management of the Service Templates and their associated SOERs. A SOER is used by the agents for the building of a Service Offer (SO) based on a ST. A SO exists only in the execution/declarative environment of an agent like the Composition Agent (CoA), the Trader Agent (TrA), and the Facility Agent (FA) itself (cf. System Environment in chapter 8). The declarative environment and its data structures like for instance the SO are elements of the Composition Model (C-Model) described in chapter 12.

The distinction between ST and SO allows the FA the implementation of availability control, resource management, and reservation of Service Offers. At first, a ST enables a FA to publish Service Descriptions without considering the current state of the resources. This might be of interest for Trader Agents, which pro-actively compose potential services. Nevertheless, the Service Description and Service Publication have to be adapted to the current availability of the resources in a second step. A SOER associated with a ST is used for this adaptation fitting to the actual situation of availability. Later after the Composition Process described in the C-Model took place, when a SO became a Candidate Service for a Service Request, the reservation of resources for the Candidate Service might become necessary, in order to ensure a deployment. The data structure of ACTAS supports a reservation on the basis of the SOER of this SO. The idea behind the use of a Service Template is partly originated in the use of Service Types in the ODP trading model (cf. [ITU1997]).

As a person can act in different roles, a service can have different Service Modes. For instance, a gateway can connect different kinds and numbers of devices. Another example is the service of a travel agency, which offers in one mode travelling by plane and in another one travelling by train. A Service Template enumerates several Service Modes, which are distinguished through discriminable sets of Semantic Characteristics.

On the side of Service Providers, it is assumed that Service Administrators exist, who are responsible for the management and description of a certain family of services, since only they can describe properly the properties and compatibilities of a service (cf. Fig. 18 - Agents Environment of ACTAS in chapter 8). A so-called "God-view" of compatibilities can hardly cope with the diverse possibilities of an actual service family and its current Service Offers. In order to describe these possibilities of a service in one Service Description, the ST wraps several Service Modes for a service with Service Ports holding as interfaces one specific set of Compatibility Characteristics each.

Example 8 **Service Administrator and Service Description**

The Service Administrators can offer the services with appropriate, possibly several Service Descriptions. However, since they do not know the current and future Service Discovery environments many points stay ambiguous: the locations of publications, the handling with different Service Descriptions, other aspects of the service besides the Service Design, and the completeness of information together with its constraints. It can happen that although a Service Description matches a Service Request, the service itself cannot be deployed since the Service Requester and the Service Provider disagree on the set of currently used Web Service standards. In Example 6, a Compatibility Characteristic was discussed for the verification of the treatment of Web Service standards in the deployment phase. Such a Compatibility Characteristic could have properties with appropriate algorithms for the negotiation of the supported standards.

The Service Description in ACTAS is like e.g. a WSML Service Description of WSMO first of all informational; only the Service Offer inside of the agents will deal with concrete data and algorithms. This is conform to the ideas of Berners Lee in the context of (Semantic) Web Services [Ber2003], who demanded that the data for automation of the Service Description has to be independent from the service behaviour and realisation. The framework character of ACTAS extends his views in the direction of linking established Service Discovery algorithms with its data, in order to achieve a more reliable automation without restricting the transparent improvement of these algorithms.

10.1 Service Templates (ST) and SOER

The structure of a Service Template is shown in Definition 8. A ST (or SO) has a Common Part and one specific part for each Service Mode (SM). Each Service Mode consists of at least one Service Port (SP). A Service Port (SP), as an interface description of the service, keeps sets of references of Compatibility Characteristics $(\mathsf{CCh}^{Ref})^{\mathsf{Set}}$. SM and the Common Part of ST hold references of General Characteristics $(\mathsf{GCh}^{Ref})^{\mathsf{Set}}$. The reference, likely a URL, is done to the ontological repositories of characteristics (cf. section 9.1), in order to ensure the use of a commonly agreed Semantic Characteristic. A Char Property is named a **Service Property**, when its Semantic Characteristic is used for the design of a Service Description.

The Service Ports with their Compatibility Characteristics form the interface of the service (in its specific Service Mode) in ACTAS. A Service Mode "coordinates" several Service Ports, i.e. the interface of a service in ACTAS can deal with several other services of ACTAS. Every Service Port can offer or request a business like service, i.e. services with client and server relationship. In this case, the ACTAS Service Mode is a real coordinator between offering and requesting of several services. However, in Technical Services with non-directed relationships like a gateway, which translate data following standard A at one Service Port to data following standard B at another Service Port, the Service Mode in ACTAS represents exactly this technical service.

Definition 8. Service Template (ST)

$\mathrm{ST} \equiv \left(\mathrm{ST}^{\mathrm{ID}}, \mathrm{FA}^{\mathrm{Ref}}, \mathrm{SM}^{\mathrm{Set+}}, (\mathrm{GCh}^{Ref})^{\mathrm{Set}}, \text{ST-Env}\right)$

$\mathrm{SM} \equiv (\mathrm{SM}^{\mathrm{ID}}, \mathrm{SP}^{\mathrm{Set+}}, (\mathrm{GCh}^{\mathrm{Ref}})^{\mathrm{Set}}, \text{ST-Env})$

$\mathrm{SP} \equiv (\mathrm{SP}^{\mathrm{ID}}, (\mathrm{CCh}^{\mathrm{Ref}})^{\mathrm{Set}}, \text{ST-Env})$

Service Template (ST) consists of:

- Name of Service Template ($\mathrm{ST}^{\mathrm{ID}}$)
- Reference of the exporting FA ($\mathrm{FA}^{\mathrm{Ref}}$)
- Set of Service Modes ($\mathrm{SM}^{\mathrm{Set+}}$)
- Set of Service Ports ($\mathrm{SP}^{\mathrm{Set+}}$)
- Set of References of General Characteristics ($(\mathrm{GCh}^{Ref})^{\mathrm{Set}}$)
- Set of References of Compatibility Characteristics ($(\mathrm{CCh}^{Ref})^{\mathrm{Set}}$)

$\text{ST-Env} \equiv (\text{Va-Co}^{\mathrm{Set}}, \text{Ex-Co}^{\mathrm{Set}}, \text{ST-Option-Slot}^{\mathrm{Set}})$

Environment of Service Template (ST-Env) consists of:

- Set of Value Constraints ($\text{Va-Co}^{\mathrm{Set}}$)
- Set of Exchange Constraints($\text{Ex-Co}^{\mathrm{Set}}$)
- Set of Option-Slots of ST ($\text{ST-Option-Slot}^{\mathrm{Set}}$)

The use of Semantic Characteristics as "building blocks" and concepts for the classification of services are discussed in section 9.1. A General Characteristic assigned to a service helps to classify the services. Such a General Characteristic could describe annotating Service Properties like location, provider, security, or other annotating non-functional parameters of the service (cf. section 2.3.2). A General Characteristic used in a Service Mode is only valid for this specific mode of the service. It also overwrites a General Characteristic with the same reference given in the common description part of the Service Description.

An environment description (ST-Env), distributed over the elements of the Service Description, keeps Value Constraints, Exchange Constraints, and Option-Slots. Option-Slots rule the interpretation of a Service Description and the Composition Process (cf. C-Model). The setting of Value Constraints enables the Service Administrator/Designer to adapt the values of the Service Properties further to the semantic context of the service. As Char Properties, the Service Properties received already their first Value Constraints for the adaptation to the semantic context of their Semantic Characteristic (cf. Definition 4.) Value Constraints use the methods of the Char Property Classes. A SOER can declare with an Option-Slot, that it "overwrites" the Value Constraints of its Service Template. In this case, the Value Constraints of the affected Service Properties, as they were listed in the environment of the ST, are discarded. An extended discussion of constraints is in section 10.4.

Definition 9. Service Offer Export Record (SOER)

$\text{SOER} \equiv (\text{SOER}^{\text{ID}}, \text{ST}^{\text{Ref}}, \text{FA}^{\text{Ref}}, \text{Time-Stamp}, \text{Valid-SM}^{\text{Set+}}, \text{SOER-Env})$
$\text{Valid-SM} \equiv (\text{SM}^{\text{Ref}}, \text{Valid-SP}^{\text{Set+}}, \text{SOER-Env})$
$\text{Valid-SP} \equiv (\text{SP}^{\text{Ref}}, \text{SOER-Env})$

Service Offer Export Record (SOER) consists of:

- Name, Identification of SOER (SOER^{ID})
- Reference of Service Template (ST^{Ref})
- Reference of Facility Agent (FA^{Ref})
- Time-Stamp contains a Start-Time and possibly End-Time of validity
- A set of valid Service Mode descriptions ($\text{Valid-SM}^{\text{Set+}}$) (at least one)
- A set of valid Service Port descriptions ($\text{Valid-SP}^{\text{Set+}}$) (at least one)
- Reference of valid Service Mode (SM^{Ref})
- Reference of valid Service Port (SP^{Ref})

$\text{SOER-Env} \equiv (\text{Va-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}}, \text{Option-Slot}^{\text{Set}})$
Environment of SOER (SOER-Env) consists of:

- Set of Value Constraints ($\text{Va-Co}^{\text{Set}}$)
- Set of Exchange Constraints (Ex-Co)
- Set of Option-Slots ($\text{Option-Slot}^{\text{Set}}$)

In the instance of a Service Template shown in Example 9, the Option-Slots are illustrated with a grey shade in Fig. 30. Compatibility Characteristics have an orange shade, whereas the General Characteristics are shown with a yellow background. They Option-Slots appear for the Common Part (beginning on the right sight), each SM, and each SP. For a simplification of the illustration, the Option-Slots contain the Ids of the elements in Fig. 30. The Option-Slot "Request" marks so-called Request Port. These are Service Ports, which can only contain Request Characteristics. With the Option-Slot "IN", the Request Ports become an interface, i.e. the server side, for Service Requests. Request Characteristics are a special kind of Compatibility Characteristics (cf. section 9.1) for the use in Service Requests. They allow an adaptation of Service Requests towards certain user groups. The Option-Slots allow declaring a direction for the Service Ports, in order to support the client-server relationship for the matching. The ST instance in Example 9 has also "non-directed" Service Ports. These are used for the description of compatibility for Component Services of e.g. Technical Service, since no client or server can be recognized, when two technical services are composed on the base of a certain communication standard like H.323. The section 10.2 discusses the Option-Slots more in detail.

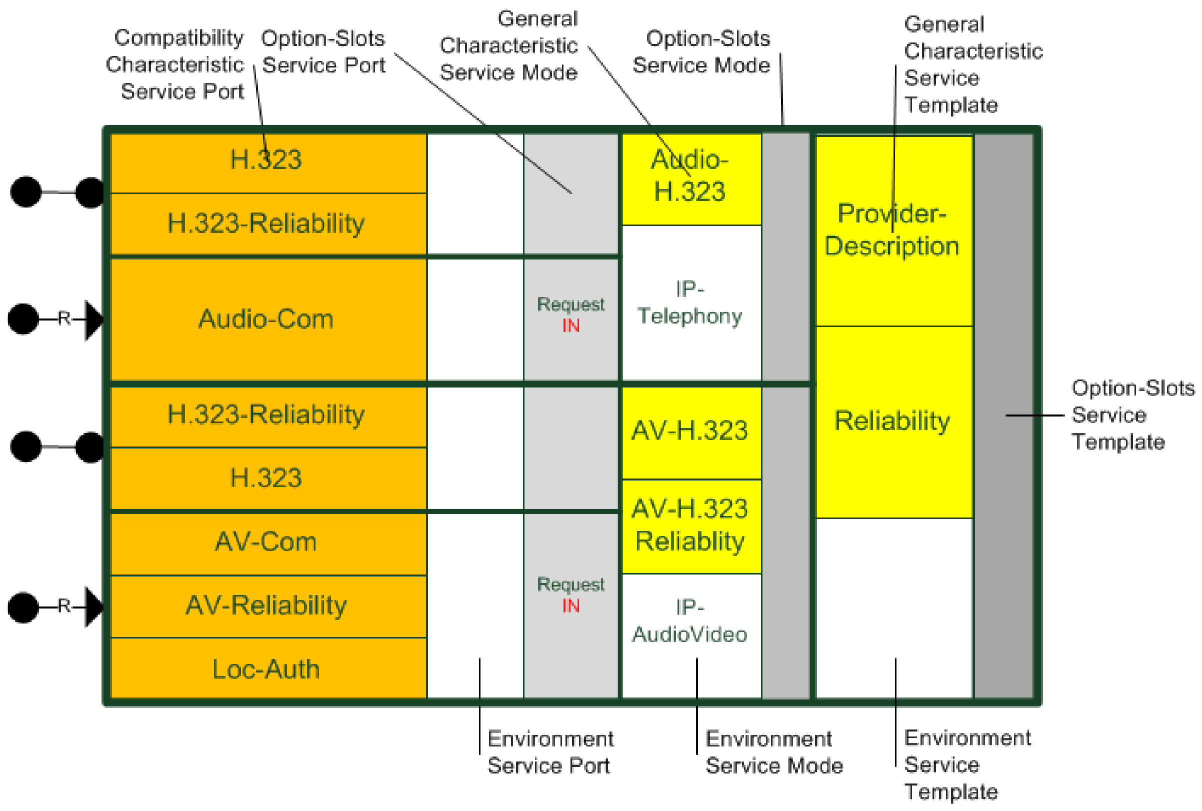


Fig. 30 - Service Template data structure

The publication of a service is done through a Service Template (ST) and Service Offer Export Record (SOER). The SOER (cf. Definition 9) adapts the Service Description of the ST according to the currently available Service Modes. Therefore, a SOER enumerates the valid Service Modes $(\text{Valid} - \text{SM}^{\text{Set}+})$ as well as their valid Service Ports $(\text{Valid} - \text{SP}^{\text{Set}+})$. The values and constraints of the Service Properties are also adapted accordingly through SOER environments. The associated ST and its FA is mentioned in the SOER. It is up to the FA to manage its ST and SOER descriptions. Only the FA can decide which SOER are valid and which SOER it accepts for the reservation of services and their needed resources for the deployment. Each reservation of resources might certainly generate the publishing of new SOERs. A SOER can anyway contain besides its identification also a Time-Stamp, in order to keep hold of its validation.

In a future version of ACTAS, a FA can directly react on Service Requests through the publication of new Service Modes dynamically extending the Service Description of a ST, when it recognizes that the semantic descriptions (cf. Definition 5) can be provided by the service. However, a selection of a Service Mode must always be based on a valid SOER, in order to allow a reservation and availability control. A SOER contains a time stamp and only the responsible FA declares, which SOER and reservations are valid.

General remark to the figures and examples in the thesis: In the introduction of the models of ACTAS, mostly simple Technical Services are schematically shown. On the one hand, it demonstrates that ACTAS is not restricted to business like services as they are mostly tackled by (Semantic) Web Services. Examples in this direction are added in the evaluation chapter. On the other hand, Technical Services contain often non-directed compositions. Thus, this supported kind of composition can be also shown. In the figures, the Compatibility Characteristics are orange, General Characteristics appear with yellow rectangles, and Service Properties have a green background, generally. Composition links have endings of a circular and arrow type. The latter, if they are directed compositions. The distinction between B2B-like and B2C-like service composition is normally illustrated with an "R" for "Request" at the drawn connection, since ACTAS supports the B2C-like service composition with Request Ports, these are Service Ports holding only Request Characteristic, a special kind of Compatibility Characteristics. The references of the Semantic Characteristics, which are listed in the definitions of the data structures, in order to state that they must be distinct and related to the ontological repository of the Semantic Characteristics, are in the examples simply meaningful names.

Example 9 **Service Template for Technical Service**

In Fig. 30, an example of a Service Template is shown. The ST consists of two Service Modes (named SM-7.1 and SM-7.2). SM-7.2 offers through one Service Port as a server (IN-SP, IN direction as Option-Slot) a service fulfilling three Compatibility Characteristics ("AV-Com", "AV-Reliability", "Loc-Auth"). All three Compatibility Characteristics were declared as Request Characteristics (RCh). The RCh "AV-Com" could be interpreted as an offered Audio-Video-Communication Service, which complies with a non-functional "AV-Reliability" characteristic. Obviously, it is an audio-video-conference facility behind this service. Therefore, it makes sense that the discovery framework checks the spatial availability and if the server client has access rights. This characteristic of the service is indicated through the third RCh: "Loc-Auth". Only a service or request with an OUT-SP (Service Port with OUT direction as Option-Slot) demanding the same three Request Characteristics is principally compatible with this Service Port (SP). The second SP of SM-7.2 is a non-directed (no "direction" Option-Slot) Service Port asking with a Compatibility Characteristic for a H.323 device, which fulfils also a H.323-reliability characteristic. The Service Mode contains two General Characteristics (GCh). One GCh translates AV Service Properties into H.323 Service Properties. Another one links the reliability Service Properties of the Service Ports. In the Common Part of the ST is one GCh holding information about the Service Provider. A GCh called "Reliability" might calculate a more general value for the reliability of the service. The other Service Mode SM-7.1 offers for the same facility only an Audio-Communication service (RCh: "Audio-Com"). SM-7.1 needs like SM-7.2 a H.323 device for the realization. The meaning of the service characteristics is fixed through the ontological repository of their publication.

10.2 The Environment Declaration

The environment declaration of the Service Description in ACTAS contains Value Constraints, Exchange Constraints, and Option-Slots. The Value Constraints describe restrictions for the Service Properties. Several Service Properties are interrelated through Exchange Constraints. The section 10.4 will take a closer look at the constraints. Each part of the Service Description (the Common Part, a Service Mode (SM), and a Service Port (SP); cf. Definition 8 and Definition 9) has its own environment description. Generally, an environment of an SM will not contain constraints for Service Properties of another SM, since the compatibility description of ACTAS will select only one Service Mode. The information of the environments will be combined in the Composition Process of the C-Model. The information of the environments of SOER can overlay the information of the environments of ST. This is controlled with an Option-Slot. In this section, possible Option-Slots are discussed.

Definition 10. Option-Slots

An Option-Slot contains information controlling the interpretation of Service and Request Descriptions as well as the processing in the C-Model. They can be defined for Service/Request Ports, Service/Request Modes, and Common Parts of the descriptions.

An Option-Slot (Definition 10) controls the interpretation of the environments and can influence the Composition Process in the C-Model. It creates an adaptability of ACTAS for alternative composition algorithms. It depends on the individual specification of an Option-Slot, in order to say if an Option-Slot, given in a Service Port or Service Mode, overwrites another Option-Slot of the same kind, given in the Service Modes or in the Common Part, respectively. The Option-Slots in the SOER environments (SOER-Env, cf. Definition 9) can overlay the Option-Slots of the ST environment (ST-Env, cf. Definition 8). Table 13 and Table 14 summarize some Option-Slot specifications for Service Ports and Service Modes, respectively.

Important for environments is the "overwrite" Option-Slot, which can appear in a SOER environment telling that the specifications in the environment of the corresponding ST part should be discarded. The Option-Slot can specify more in detail, which part of the ST environment should be discarded: the whole environment, all Option-Slots, all constraints or specific constraints.

The "request" Option-Slot declares a Service Port as a so-called **Request Port**. A Request Port must only contain Request Characteristics as Compatibility Characteristics and is used in the context of Service Requests (C-Model). With the set of Request Characteristics, a Request Port is semantically associated with a specific user group (the effective user group is defined through the intersection of the user groups specified in the semantic descriptions of the Request Characteristics (cf. section 9.2) being elements of the set.).

The Service Ports can be declared as directed or non-directed, which is done through the "direction" Option-Slot described in Definition 10 only for the Service Ports (including Request Ports). Technical Services have often a non-directed relationship between Component Services. In a Client-Server relationship of compatible services, a declaration of the direction in the compatibility description is necessary, in order to support Service Compositions for workflows. Thus, a Service Port of a service, which requests another service, has in the "direction" Option-Slot the attribute OUT. Alternatively, a Service Port, offering a service (i.e. being a server) has the attribute IN. The matching algorithm of a Merge Constraint takes later advantage from this direction declaration. Due to the fact that OWL-S Service Description do not distinguish between goal and service like WSMO Service Descriptions, this Option-Slot is necessary for the

Option-Slots	Semantic and Attributes
Direction	Declaration of a Server-Client relationship of Service Ports: • *direction*(OUT) – Client Port, "OUT Port" • *direction*(IN) – Server Port, "IN Port"
multi-port	The Service Mode can have several Service Ports of this kind. Depending on the Composition Process the Service Port will be cloned: • multi-port(max) – Attribute max defines the maximum number of cloning allowed.

overwrite
overwrite(VALUE)
overwrite(EX)
overwrite(OPTION)
overwrite(VALUE, *list-of-properties)*
	"Overwrite" is an Option-Slot appearing only in a SOER specification. All specifications in the environment (Env) given in the correspondent ST Service Port are discarded. A list of property references could specify more closely that only the Value Constraints of certain properties or other kinds of specification should be discarded.

Request	A Service Port can be declared as a **Request Port**, i.e. all Compatibility Characteristics in the Service Port must be in fact Request Characteristics
facility-agents(list-of-FAs) trader-agents(list-of-TrAs)	Enumeration of agents, which should be preferentially asked for compatible Service Offers in the composition or Trading Process. In a closer specification, it could be expressed if these agents are compulsive.

Table 13 - Option-Slots of Service Ports

distinction between client and server side and the fitting interpretation of the information. In the case of a WSMO Service Description, Service Port holding the OUT-direction (**OUT Port**) will keep the WSMO goal, whereas the **IN Port** will contain the WSMO service specification.

The Option-Slots can control the Composition Process. For instance, the algorithm of a CoA can implement that the OUT Ports of a selected Service Mode will be only considered, when all IN Ports are matched with (principally) compatible Service Ports. In this way, loops in the Composite Structure can be avoided and some resulting constraints of the IN Ports could be exchanged with the OUT Ports, in order to make the Composition Process more selective in an earlier state.

Several Option-Slots influence the interpretation of Exchange Constraints (cf. section 10.4.3 and Definition 18). Two of them are mentioned in Table 14: "translation" and "exchangeProperties". The "translation" Option-Slot is interpreted in the Composition Process of the C-Model. The time point of the application of the Exchange Constraints depends on the realised composition algorithm in the CoA. Normally (at least in the introduced Composition Process in the C-Model description), the Exchange Constraints are resolved, when a candidate for the whole Composite Service is found based on principal compatibility and the application of the Merge Constraints, because then a relative maximum of information for the application of

Option-Slots	Semantic and Attributes
translation translation ($(\text{Ex-Co}^{\text{Ref}})^{\text{set}}$)	Early interpretation of Exchange Constraints The meant Exchange Constraints can be listed.
exchangeProperties ($\text{Ex-Co}^{\text{Ref}}$, ExchangeProperties)	Adaptation of the Service Properties used by Exchange Constraints. This Option-Slot overwrites the term "ExchangeProperties" of the referenced Exchange Constraint ($\text{Ex-Co}^{\text{Ref}}$). (The reference is similar the reference of a property (cf. Definition 15) with the exception of having an Ex-Co^{ID} (cf. Definition 18).) This is useful, when Exchange Constraints are imported e.g. through General Characteristics as "building blocks" (cf. Example 15)

overwrite
overwrite(VALUE)
overwrite(EX)
overwrite(OPTION)
overwrite(VALUE, *list-of-properties)*
	Like in Definition 10 only valid for the greater scope of a Service Mode or Common Part, respectively
	VALUE – Value Constraints
	EX – Exchange Constraints
	OPTION – The Option-Slots

Table 14 - Option-Slots of Service Modes and Common Part of Service Descriptions

the Exchange Constraints is available. With the "translation" Option-Slot set, the Exchange Constraints will be applied earlier in the Composition Process, likely when possible (Component) Services are just found and composed. In this way, values of the Service Properties in the just composed Service Ports (probably IN ports) can be translated through data mediation into values of Service Properties in the Open Ports. In this way, the values in the Open Ports are already quite defined for an extended Service Discovery, which involves the solving of Merge Constraints at the side of the Trader or Facility Agent. The "translation" is used, in order to mediate between different interfaces of a service. The evaluation has a closer look at the use of Translation Offers.

Other Option-Slots can select Facility Agents for the composition (e.g. "facility-agents" in Table 13 - Option-Slots of Service Ports). Such a Selection might come handy; when potential Facility Agents offering the request service are known or learnt through positive feedbacks (cf. System Environment, phase 6, section 8.6 and future research).

10.3 Description of compatibility

The S-Model has to describe two kinds of compatibilities of a service. On the one hand, a service can be compatible to another service. On the other hand and more important, some services have to be compatible to Service Requests (R-Model, chapter 11). Service Request and the Service Description of the S-Model are transformed into data structures of the declarative environment of the C-Model (chapter 12). These data structures are used for the determination

Definition 11. Principal Compatibility for services and Service Ports

A service$_A$ is **principally compatible** with a service$_B$ iff
they have principally compatible Service Ports SP$_A$ and SP$_B$.

Two Service Ports SP$_A$ and SP$_B$ are principally compatible iff
they hold the same set of references of Compatibility Characteristics ($(\mathrm{CCh}^{\mathrm{Ref}})^{\mathrm{Set}}$ in Definition 8) as interface and compatible sets of Option-Slots (cf. Definition 13).

The principal compatibility of the Service Ports (SP$_A$ and SP$_B$) leads to the selection of their Service Modes SM$_A$ and SM$_B$ on service level and the association of comparable Service Properties on the level of Service Properties (cf. Definition 12).

of principally compatibility between Service Offers as well as between Service Offer and Service Request. Subsequently, the compatibility of the compositions is further tested through the rules of phase 4, testing whether they fulfil the Merge Constraints.

In the S-Model and R-Model, the principal compatibility description is done through a set of Compatibility Characteristics at the Service Ports. (**Service Port**s describing the compatibility between Service Requests and Service Offers are called **Request Port**s (**RP**), since they have as Compatibility Characteristics only Request Characteristics.)

The Compatibility Characteristics hold the Merge Property Classes, which are used for the building of Merge Constraints. The Merge Constraints as well as the principal compatibility can be controlled through Option-Slots. Only Service Ports with compatible Option-Slots can be composed in the Composition Process.

Definition 12. Comparable Service Properties

Two Service Properties of principally compatible Service Ports are called "comparable" iff

- They have the same names.
- They are declared in the context of Compatibility Characteristics with the same references

Definition 13. Compatible sets of Option-Slots

Let be $\text{ST-Option-Slot}^{\text{Set}}{}_A$ and $\text{ST-Option-Slot}^{\text{Set}}{}_B$ sets of Option-Slots appearing in the environments of two Service Ports SP$_A$ and SP$_B$. Empty sets of Option-Slots of Service Ports are always compatible for composition. Further compatible sets of Option-Slots are shown in Table 15

Especially the "direction" Option-Slot (cf. Table 13) must be observed, since it describes a Service Port as being on the server/Service Provider side ("IN Port") or at the client side ("OUT Port"). Definition 13 declares that only Service Ports with no "direction" Option-Slots or with "direction" Option-Slot holding alternating attributes are compatible. The set of Compatibility

Option-Slots	Compatibility Rule
The sets of Option-Slots are empty.	The sets of Option-Slots are compatible.
Request	Either both set have the "request" Option-Slots (Request Ports) or none of them.
Direction	The ports hold compatible "direction" Option-Slots iff $(direction(\mathrm{OUT}) \in \text{ST-Option-Slot}^{\text{Set}}_{A} \wedge direction(\mathrm{IN}) \in \text{ST-Option-Slot}^{\text{Set}}_{B}) \vee$ $(direction(\mathrm{IN}) \in \text{ST-Option-Slot}^{\text{Set}}_{A} \wedge direction(\mathrm{OUT}) \in \text{ST-Option-Slot}^{\text{Set}}_{B})$

Table 15 - Compatible sets of Option-Slots (cf. Table 13)

Characteristics of an IN Port describe criteria of the offered service. Analogically, the set of Compatibility Characteristics of an OUT Port describe the criteria of a requested service. In Example 9, the service is offered for Service Requests in two ways (two Service Modes). Since it is an interface between Service Request and the described service, it is done through a Request Port declared as IN Port.

Example 10 **Principal Compatibility with Comparable Properties**

The Fig. 31 shows a principal compatibility with and without a direction. In the directed compatibility relation, two Request Ports (RP) are associated, which combine two Request Characteristics, in order to look for a travel and insurance service. The Technical Service delivered with two phones does not have any directions, since the phone services do not distinct between server and client. The described interface of the Technical Service is transparent for the Service Clients (cf. Example 2). This is reflected in ACTAS, since the Service Ports with the Compatibility Characteristics are not declared as Request Ports. Since principally compatible services hold the same set of Compatibility Characteristics, the Service Properties become "comparable" (cf. Definition 12). This Example is

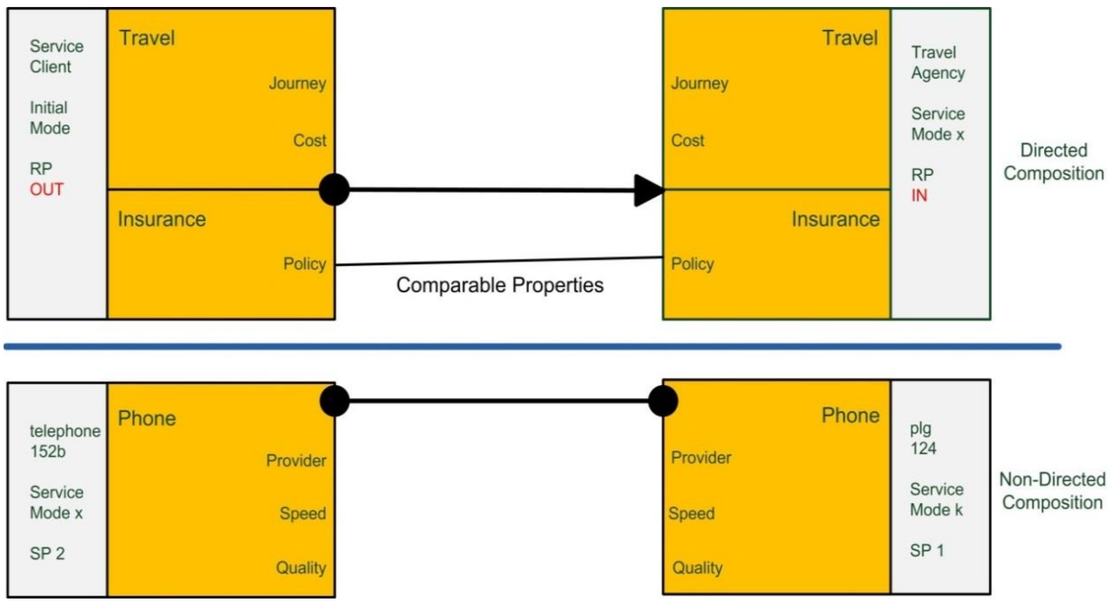


Fig. 31 - Principal Compatibility for directed and non-directed composition

extended through Example 13 for the discussion of Merge Constraints.

Technical services can describe compatibility for Service Ports without a direction (i.e. no "direction" Option-Slot). For instance, a connection, built on the base of a certain standard between to technical devices, has no directions. Therefore, there is no "direction" Option-Slot at the Service Ports describing a necessary H.323 connection in Example 9. In Example 10, the principal compatibility description is shown again for a directed and non-directed relation.

ACTAS ensures with the "Request" Option-Slot that only Request Ports, i.e. Service Ports containing entirely Request Characteristics, can be composed (cf. Table 15). The "facility agents" Option-Slot (cf. Table 13) might only accept SOERs (and therefore STs) published by the mentioned Facility Agents. It becomes more complicated to fulfil the "trader agents" Option-Slot. This can only be done through the CoA.

10.4 Constraints in the Service Description

Several kinds of implicit or at least unchangeable constraints are integrated in the S-Model. First of all, ontological schemas define constraints. They are applied in the ontological repositories of the Property Classes and Semantic Characteristics. Additionally, the algorithmic interpretation of the Service Properties realised through the Property Classes allows the integration of application and domain ontologies. Secondly, implicit constraints are incorporated in the pro-active behaviour of the agents, since the traders, application environments and Service Providers have their own policies. Thirdly, the integration of external algorithms through the Property Classes leads to transparent Attribute Constraints. The Service Designer has no control over these Attribute Constraints as discussed in section 9.2. Fourthly, he accepts the constraints of a Semantic Characteristic, i.e. its environment and Char Properties (cf. Definition 4) as well as its semantic description (cf. Definition 5), as soon as he uses it as a building block of a Service Description. Very important are here the Merge Constraints defined through the association of Merge Property Classes with the properties of Compatibility Characteristics. The Merge Constraints cover matching and potential mediation.

However, the Service Designer can start to formulate specific constraints in the environments of the Service Templates and the Service Offer Export Records. These are Value Constraints and Exchange Constraints. In a dual way, the Service Requester can give constraints in the environments of a Service Request (cf. R-Model). Later on, the composition and Trading Process of the C-Model uses the constraints for the selection of Service Candidates. Additionally, the Service Designer can (partly) decide, which Service Property (objects) are involved in the Exchange Constraints gained with General Characteristics as a building block of the Service Description. Service Designer as well as Service Requester use references to Service Properties and the methods listed in the interfaces of the Property Classes ($\mathsf{Method}^{\mathsf{Set}}$, cf. Definition 6).

In a later phase of the life cycle, the negotiation between the Facility Agents can help to overcome coordination constraints of Component Services. However, the solving of these constraints is out of scope of ACTAS as long as they are not partly tackled in the context of a used Semantic Characteristic. This is possible, since a Semantic Characteristic can be semantically connected with the later phases of the life cycle, in order to clarify for instance the methods of coordination. The feedback can lead to some learnt constraints in later versions of ACTAS.

ACTAS distinguishes three different kinds of constraints: Value Constraints, Merge Constraints, and Exchange Constraints. In the following sub-sections these kinds of constraints are described more in detail. In section 9.2, three different kinds of Property Classes were defined (Definition 7), which allowed the access of possibly external algorithms.

In the declarative environment of ACTAS, built in the C-Model, the properties are comparable to classes in object-oriented languages. They contain methods for editing and comparing of values. The methods of Property Classes are used for the setting of (internal) Value Constraints or the creation of new output objects from the input object (cf. rule (ExOp)).

10.4.1 Value Constraints

The **Value Constraint**s (**Va-Co** (Definition 14) restrict the possible values of a Service Property through the application of methods offered by its Char Property Class (cf. Definition 6). This Char Property Class was used for the declaration of the Service Property as a Char Property in the context of its Semantic Characteristic (cf. Definition 4). In an ideal case, the method just set a distinct value for the Service Property.

Definition 14. Value Constraint (Va-Co)

$\mathrm{ValueConstraint} \equiv \text{va-co}(\mathrm{Property}^{\mathrm{Ref}}, \mathrm{ValueClause})$

$\mathrm{ValueClause} \equiv [\mathrm{clause}_1, \cdots, \mathrm{clause}_m]$
$\mathrm{clause} \equiv \mathrm{method}_x([Parameter_1, \cdots, \mathrm{Parameter}_n]),$
with $\mathrm{method}_x \in \mathrm{method}^{\mathrm{Set}}$ of $\mathrm{CharPropClass}^{\mathrm{Ref}}$ ($cf. PC$, Definition 6)

Meaning:

- $n, m \in \mathbb{N}$,
- $\mathrm{Property}^{\mathrm{Ref}}$ is a reference to the Service Property (cf. Definition 15)
- The ValueClause contains an conjunction of methods calls (i.e. semantically ($\mathrm{clause}_1 \wedge \cdots \wedge \mathrm{clause}_m$))declared in $\mathrm{method}^{\mathrm{Set}}$ of the Property Class($\mathrm{CharPropClass}^{\mathrm{Ref}}$) of the Service Property
  (each method_x will be translated in an individual op/1 predicate later)
- $\mathrm{CharPropClass}^{\mathrm{Ref}}$ as a reference to the Property Class (PC) origins from the declaration of the Service Property in the context of its Semantic Characteristic (cf. Definition 5), which is referenced through Char^{Ref} in $\mathrm{Property}^{\mathrm{Ref}}$.

In a Va-Co, the affected Service Property is clearly referenced ($\mathrm{Property}^{\mathrm{Ref}}$ in Definition 15). Definition 15 describes this reference more closely. It contains the identification of the description part of the Service Property (e.g. a Service Mode of a Service Template ($\mathrm{SM}^{\mathrm{ID}}$ in

Definition 8) or a Request Port of a Service Request ($\mathrm{RP}^{\mathrm{ID}}$ in Definition 19)), the addressing of the Semantic Characteristic, and the name of the Service Property.

Example 11 **Value Constraints**

In Example 7, Principal Char Property Class, the existence of a Property Class dealing with OWL-S-capability descriptions was assumed. The initialisation of a Service Property could be done through a Value Constraint with a parameter containing a URL pointing to an OWL-S Service Description. That means, that in one environment description (Env) of the Semantic Characteristic or of the Service Description (e.g. in the ST-Env of a Service Mode) a Value Constraint exists for this Service Property exist. A Service Property for temperatures in the semantic context of room location describing characteristic could get the Value Constraints in the Char-Env that the temperature range should be between 15 and 23 degree Celsius.

Definition 15. Reference of a Service Property

$\mathrm{Property}^{\mathrm{Ref}} \equiv \text{prop-ref}(part^{Ref}, \mathrm{Char}^{Ref}, \mathrm{Name})$

Meaning:

- $part^{Ref}$ is a reference to the description part, e.g. $\mathrm{ST}^{\mathrm{ID}}$ for the Common Part of a Service Template or $\mathrm{SP}^{\mathrm{ID}}$ for a specific Service Port (cf. Definition 8)
- $\mathrm{Char}^{\mathrm{Ref}}$ is a reference to the Semantic Characteristic (cf. Definition 4)
- Name of the property as given in its declaration

Value Constraints are part of the environment descriptions (Env) in the Semantic Characteristics and the Service Descriptions, i.e. each element of the S-Model (and R-Model) can define Value Constraints. In the C-Model, the Value Constraints of one Service Property are collected in one ordered set interpreted as a big conjunction of all clauses. The Value Constraints, given in the environment of a characteristic (Char-Env in Definition 4) have the highest priority. The discussion about Option-Slots (cf. section 10.2) showed that Value Constraints in a SOER environment can overwrite the ones in the ST environment, i.e. the Value Constraints of the ST are simply discarded in the collection process of the C-Model in this case.

10.4.2 Merge Constraints

For the "merge" of Service Properties, an extended test of Service Compatibility on the level of the comparable Service Properties (cf. Definition 12), takes place. Comparable Service Properties are declared as Char Properties in the semantic context of the **same** Compatibility Characteristic, this means they have associated a common **Merge Property Class**. This Merge Property Class contains the "merge" constructor which takes as input the comparable Service Properties and the "direction" Option-Slot. The "direction" Option-Slot (cf. Definition 10, Table 13 - Option-Slots of Service Ports) is part of the Service Port environment (ST-Env in Definition 8). The construction of Merge Property Object in the C-Model with the "merge" constructor is called an application of a **Merge Constraint**.

Example 12 **Idea of Merge Constraint**

A process model gives a detailed description of the choreography of a service's messages (cf. Fig. 9). A Service Administrator could introduce a Compatibility Characteristic with a Semantic Description that links this CCh with commonly agreed methods of choreography (4th aspect, grounding phase) in a certain application domain. This CCh could have a Char Property that held possible processes of a service. When the principal compatibility of two services holding the introduced CCh is recognized, then the two comparable Service Properties will contain possible processes from the client view and from the server view, respectively. The merge constructor of the associated Merge Property class could realize an algorithm, which checks the processes of the comparable Service Properties. The resulting Merge Constraint will only accept services as matching, when these comparable Service Properties contain a complementary process fulfilling the given process model of the application domain associated with the Compatibility Characteristic.. This means that a process in the Service Property on the client side fits to a process in the comparable Service Property on the server side. The "direction" Option-Slot helps to distinguish the server and client side.

Definition 16. Merge Constraint (Me-Co) in S-Model

A Me-Co in the S-Model is implicit, i.e. it is given through the association of a Service Property with a Merge Property Class ($\text{MergePropClass}^{\text{Ref}}$ in Definition 4) in its declaration as a Char Property of a Compatibility Characteristic (cf. Definition 4). It becomes explicit in the C-Model.

Merge Constraints (Definition 16) check if the values of two comparable Service Properties can be matched. The merging might also include mediation, when the values are based on different ontologies. Through the declaration in the context of Compatibility Characteristics, the Merge Property Classes are linked with a Service Property (cf. Definition 4). The Example 12 illustrates further the idea of a Merge Constraint. In Example 13, the Example 10 is extended. The resulting Merge Property Objects in the C-Model are shown.

Example 13 **Merge Constraints (Me-Co)**

This example extends the Example 10. Two Service Properties describe the "Travel" service: "Journey" and "Cost". The "Insurance" service is just described through its "Policy". Since the compatibility in ACTAS demands the same set of Compatibility Characteristics (in this case Request Characteristics), the Service Properties get "comparable" (cf. Definition 12). Comparable Service Properties hold the same Merge Property Class, i.e. they have associated algorithms for the matching and potential mediation, which fit to the context of the Compatibility Characteristic. The Service Properties, which became "comparable" through the principal compatibility in Fig. 31 get tested for matching through the algorithms in the Merge Property Class constructing a Property Merge Object (cf. Fig. 32). The direction of the compatibility relation has to be observed, since the Service Property "Journey" on the Service Requester side will contain the wishes, which have to be conciliated with the resources/values offered by the Server Provider side. A matching algorithm in the case of the Technical Service provided by the Phone Facilities has to find a line speed and Quality-of-Service (QoS) supported by both communication facilities. Probably, the Service Property "Provider" will not need a matching algorithm at all. Thus, it is possible to have no Merge Property Class associated with a Service Property.

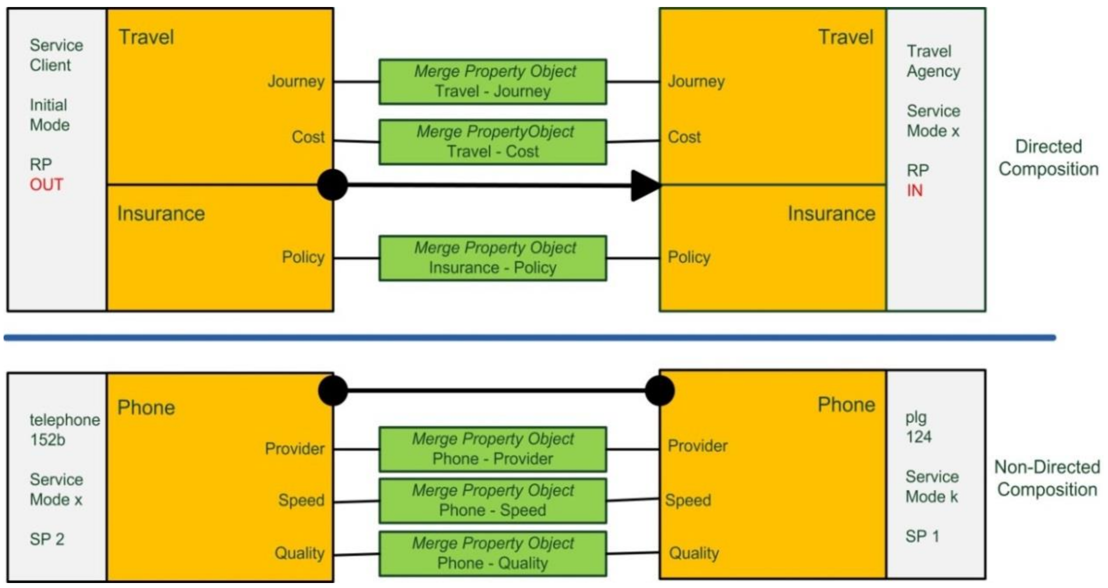


Fig. 32 - Me-Constraints for directed and non-directed composition

The access of the implementation instance of the Property Class through an object is part of the C-Model (cf. chapter 13). A Merge Property Object realises the access of information of an implementation instance, which were constructed on the base of the Merge Property Class of a Merge Constraint between two comparable Service Properties (cf. Definition 12 and Definition 16). It is a result of the extended testing of compatibility through the Merge Constraint. A Merge Property Object can be accessed in three different ways depending on the view on its data: an access of the values from (1) the client side, (2) the server side, or (3) from a merged point of view. A client will ask for a greater range of values than the server can provide.

It is the Merge Constraint algorithm, connected with the Service Property in the context of its Compatibility Characteristic, which determines if there can be a compromise between the interest of the client and the server, i.e. a matching. However, it might be still of interest to look at this matching again from this different perspective. For instance the "Cost" Service Property of the "Travel" Compatibility Characteristic in Fig. 33 could still deliver the requested costs for the travel (client view), the offered costs (server view), and the matched costs (merged view). The

different views also make sense for a non-directed merge. The "Quality" Service/Char Property of the "Phone" Compatibility Characteristic determines the quality of the connection between the two phone facilities. The Merge Constraint determined a level of quality, which can be supported by both of them. Looked from their Service Ports at the Merge Property Object, the Merge Property Object could still provide the supported levels of quality of each of them.

The view on the Merge Property Object can be seen as an additional criterion for the property reference defined in Definition 15. It is used in Exchange Constraints, in order to get an access to a Char Property Object with the right view, i.e. the Merge Property Object must "export" an appropriate object. After the application of an Exchange Constraint, the possibly changed and according the constraints adapted Char Property Object still represents its specific view on the Merge Property Object. This means that it has to be "imported" again, in order to see, if the

Merge Constraints are still satisfied. The handling of objects is further described in the C-Model (cf. chapter 13). In the next section, a closer look on the definition of Exchange Constraints is done. The Example 15 includes the view references.

Example 14 **View on Merge Property Object**

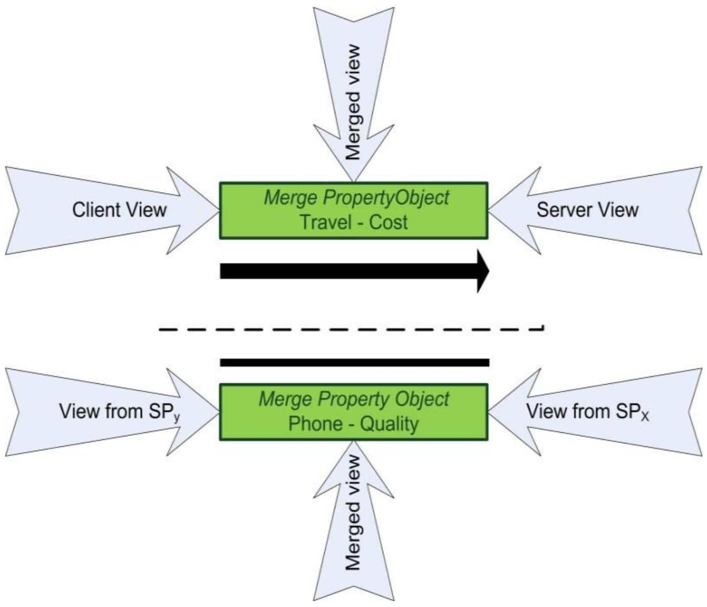


Fig. 33 - The different views on a Merge Property Object

In this example, a "Transport" Compatibility Characteristic is assumed. It has a Char Property holding the information about the numbers of items, which should be transported as well as the kind and the number of loads necessary for their transport. The Service Property defined by this Char Property could wish for the transport of n items on the client side. The kind and number of loads might be not specified on the client side. A Comparable Property on the server side could only offer truck loads with less capacity m (m < n). Obviously, the Merge Constraint can easily find a compromise. It defines as many truck loads as necessary, in order to transport the n items. Now there are three ways to look at the Merge Property:

(1) The client's point of view: n items should be transported, (2) the server's perspective: have truck loads for (m < n) items each, (3) merged perspective: k truck loads with m items plus one load with (l < m) items results in the transport of n items altogether

Definition 17. View on Merge Property Object

The information stored in the implementation instance handled through a Merge Property Object can be accessed with three different views.

- The views depend on the kind of composition tested by the Merge Constraint. Possible views for directed and non-directed compositions are shown in Fig. 33.
- The views on the Merge Property Object can be considered as an addition to the reference of a Service Property described in Definition 15.
- Proposed are the atoms "client", "server", and "merged" for a directed composition.
- A Merge Property Object of a non-directed composition could add the identification of the accessing Service Port, in order to clarify the view. Generally, the atom "spView" is proposed.
- For Service Properties, which are not part of a Merge Property, the view term is ignored.

10.4.3 Exchange Constraints

Exchange Constraints are constraints involving several Service Properties. They allow the description of relationships between Service Properties of different Semantic Characteristics and parts of the Service Description. For this purpose, they use methods of the Exchange Property Classes (referenced through ExchangeClasses) and possibly methods of Char Property Classes. The latter in order to test conditions of just one Service Property. A Char Property Class can be accessed through the reference of its Service Property ($\text{Property}^{\text{Ref}}$ in ExchangeProperties, cf. Definition 18 and Definition 15), since every Service Property was declared as a Char Property in the context of a Semantic Characteristic. Often, such tests are also implemented in the Exchange Property Classes. However, the application of every method also means the solving of constraints. The solving of constraints should be monotone, i.e. the resulting restrictions of the values should be kept. ACTAS distinguishes between tests and the application of methods. The former are not monotone, they are just valid for the time point of the test. Methods and tests are used for the realisation of the clauses of the constraints.

Definition 18. Exchange Constraint (Ex-Co) in S-Model

$$\text{ExchangeConstraint} \equiv \text{ex-co}(\text{Ex-Co}^{\text{ID}}, \text{ExchangeElements}, \text{ExchangeClause})$$

$$\text{ExchangeElements} \equiv (\text{ExchangeProperties}, \text{ExchangeClasses})$$
$$\text{ExchangeProperties} \equiv$$
$$\text{properties}\big([(\text{Property}_1^{\text{Ref}}, View_1, \text{ExName}_1), \cdots, ((\text{Property}_o^{\text{Ref}}, View_o, \text{ExName}_o)]\big)$$
$$\text{ExchangeClasses} \equiv \text{classes}\big([\text{ExClass}_1^{\text{Ref}}, \cdots, \text{ExClass}_p^{\text{Ref}}]\big)$$

$$\text{ExchangeClause} \equiv \begin{pmatrix} [\text{lh-clause}_1, \cdots, \text{lh-clause}_n] \Longrightarrow \\ [\text{rh-clause}_1, \cdots, \text{rh-clause}_m] \end{pmatrix}$$
$$\text{lh} - \text{clause} \equiv \top \mid \text{cl}$$
$$\text{rh} - \text{clause} \equiv \top \mid \text{cl}$$
$$\text{cl} \equiv \text{method}_x([\text{ExName}_1, \cdots, \text{ExName}_k], [\text{Parameter}_1, \cdots, \text{Parameter}_l]),$$
$$with\ \text{method}_x \in \text{method}^{\text{Set}} \text{ of } \text{ExClass}_y^{\text{Ref}} \text{ or of CharPropClass}^{\text{Ref}}$$

Meaning:

- $k, l, n, m, o, p \in \mathbb{N}$
- Ex-Co^{ID} identification of the Exchange Constraint
- $\text{Property}^{\text{Ref}}$ is a reference to a Service Property (cf. Definition 15)
- View is relevant for the access of a Merge Property Object (client, server, merged).
- $\text{ExClass}^{\text{Ref}}$ is a reference to an Exchange Property Class
- ExchangeProperties references the involved Service Properties of the Ex-Co and associates them with a name used in the Ex-Co (ExName)
- If lh-side ($\text{lh-clause}_1 \wedge \cdots \wedge \text{lh-clause}_n$) of ex-clause is $\top$, then
  - rh-side ($\text{rh-clause}_1 \wedge \cdots \wedge \text{rh-clause}_m$) must be valid unconditionally
  - otherwise: lh-side states a condition for the validation of rh-side

ACTAS supports the formulation of premise conditions (conjunction of clauses on the left-hand side, list of lh-clauses in Definition 18) for the testing of the clauses on the right hand side (rh-clauses in Definition 18). For the calculation of new values, it is normally demanded that a Service Property has a distinct integer or real value. An example in this direction is the is/2-predicate used for calculation in Prolog, which will only function, when its expression is ground, i.e. all the variables in the expression are instantiated with numeric values. In Example 15, the pre-condition checks if a Service Property has a value, before a method correlates the values of the three Service Properties.

An Exchange Constraint in the S-Model/R-Model (cf. Definition 18) consists of two parts: the naming of the involved elements (ExchangeElements) and the clause(s) of the constraint itself (ExchangeClause). The addressed elements are the Service Properties (ExchangeProperties) and the Exchange Property Classes (ExchangeClasses) used in the Exchange Constraint. Inside of the constraint, the Exchange Constraint uses specific symbols or names, the so-called ExNames. Through the "ExchangeProperties" term, the ExNames are associated with Service Properties. The Service Properties are accessed through references ($\mathsf{Property}^{\mathsf{Ref}}$, cf. Definition 15) and the view on the Merge Property Object (cf. Fig. 33).

In the term "ExchangeClasses" the Exchange Property Classes are referenced through URIs to the ontological repositories of the Property Classes are given (cf. section 9.2 - S-Model: Property Classes). The Property Classes themselves have the Grounding Description (cf. Definition 6), which are used in the C-Model to define Exchange Property Objects for the access of the methods and tests of the Exchange Property Classes used in the Exchange Constraint.

In Example 15, an assumed Exchange Property Class called "audioQuality-ex" was simply addressed through its name (cf. equation (10-2)). This example shows nicely the use of a General Characteristic as a "building block" of a Service Definition (cf. section 9.1). The need for the translation of the values of Service Properties held at one Service Port and the ones held at another one does often arise. Therefore, so-called Translation Offers are discussed in the evaluation. In the example, the Service Property "Audio-Quality", which is defined in the Compatibility Characteristic "Audio-Com", has to be translated into the Service Properties "Speed" and "Quality" of the Compatibility Characteristic "Phone" and vice versa. The equation (10-2) shows an Exchange Constraint, which could be used for the translation at least in one direction, since the term "ExchangeClause", which describes the rule of the Exchange Clause, demands a value for the Service Property "Audio-Quality" in its premise. The Service Properties are correlated in the conclusion. The method call has the parameter "max", which could mean, that the maximal possible speed and quality should be demanded. This idea is supported by the view references to the Merge Property Objects: "Audio-Quality" is viewed from the server side, i.e. what can be offered and not what is whished.

"exchangeProperties" Option-Slot	$\mathit{exchangeProperties}((\text{Audio-Phone}, Ex_{Co_1}),$ $\mathit{properties}([\ (\text{prop-ref}(SP_1,\ \mathit{Audio},\ \text{Audio-Quality}),\ \mathit{server},\ AQ),$ $(\text{prop-ref}(SP_2, \mathit{Phone}, \mathit{Speed}), \mathit{spView}, S),$ $(\text{prop-ref}(SP_2, \mathit{Phone}, \mathit{Quality}), \mathit{spView}, Q)])$	(10-1)
Resulting Exchange Constraint in CompSt	$\text{ex-co}\Big(Ex\_Co_{new}^{ID},$ $\Big(\text{properties}([(\text{prop-ref}(SP_1, \text{Audio-Com}, \text{Audio-Quality}), \text{server}, AQ),$ $(\text{prop-ref}(SP_2, \text{Phone}, \text{Speed}), \text{spView}, S),$ $(\text{prop-ref}(\text{SP}_2, \text{Phone,Quality}), \text{spView,Q})]),\ \mathit{classes}([\mathit{audioQual}])\Big),$ $([\mathit{audioQual}.\mathit{hasValue}([AQ],[\ \ ])] \Rightarrow$ $[\mathit{audioQual}.\mathit{translation}([AQ,S,Q],[\mathit{max}])])\Big)$	(10-2)

Example 15 **Translation Exchange Constraint**

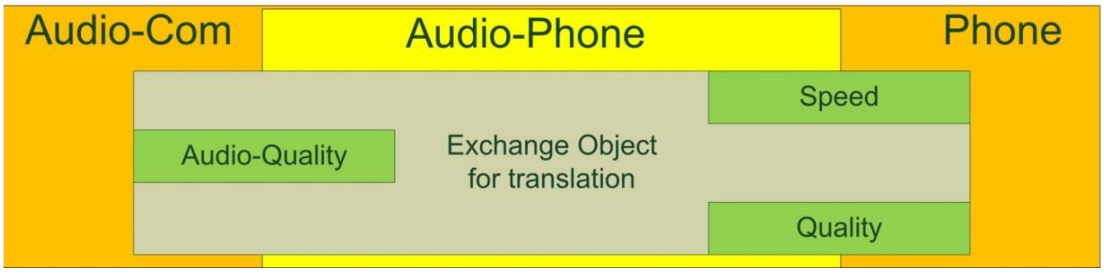


Fig. 34 - Exchange Constraint with General Characteristic

In Fig. 34, a Service Mode with two Service Ports is schematically shown which could be used for offering an audio communication service realized through a phone facility. One Service Port holds the Request Characteristic "Audio-Com", the other one the CCh "Phone". The Service Mode itself has the General Characteristic "Audio-Phone". It can be assumed that the ontological repository of the Semantic Characteristic contains "works-with" relationships between the General Characteristic and the two Compatibility Characteristics, since the General Characteristic shall hold an Exchange Constraint, which translates between the values of the Char Properties in the Compatibility Characteristics. Then this General Characteristic can be used as a building block between the two Service Ports holding the Compatibility Characteristics. Assuming further the existence of an Exchange Property Class called "audioQuality-ex" with a method "translation", which translates between the Char Properties and is applied in the Exchange Constraint of the General Characteristic, then the Service Designer would just have to adapt the "ExchangeProperties" (cf. Definition 18) in order to achieve an Exchange Constraint similar to the one shown in equation (10-2). The Exchange Constraint does specify the pre-condition that the Service Property "Audio-Quality" should have a value. This pre-condition shall prevent an early application of the Exchange Constraint.

Since the Service Description lists only references to the Semantic Characteristics (cf. Definition 8, Service Template (ST)), the "exchangeProperties" Option-Slot (cf. Table 14) takes over the task of adaptation. It overwrites the "ExchangeProperties" term of the referenced Exchange Constraint. From this point of view, the equation (10-2) can be seen as a result of an Exchange Constraint imported through the General Characteristic and an adaptation through an "exchangeProperties" Option-Slot in equation (10-1).

It is worthwhile to mention that in the assumed scenario of Example 15 the Service Designer does not have to formulate the Exchange Constraint from the sketch. He can take advantage of a General Characteristic called “Audio-Phone” (cf. equation (22-3) in Table 32 - General Characteristics (GCh)) as a “building block”, since it already contains the Exchange Constraint. However, he has to add an “exchangeProperties” Option-Slot (cf. equation (10-1)) for the adaptation of the term “ExchangeProperties” in the imported Exchange Constraint, in order to have the right associations for the Exchange Name (ExNames) of the constraint.

ACTAS - REQUEST MODEL (R-MODEL)

11 Request Model (R-Model)

In the previous chapters, the Service Model (S-Model) was introduced. Based on Semantic Characteristics and Property Classes, it enables a Service Designer to describe the services offered by a Service Provider. The perspective is changed to the Service Requester or Service Clients with the Request Model (R-Model). Apart from this change of the describing perspective, the R-Model can be seen as an extension of the S-Model, since it is based on the same "building blocks": Semantic Characteristics and Property Classes. Therefore, the introduction can be kept short.

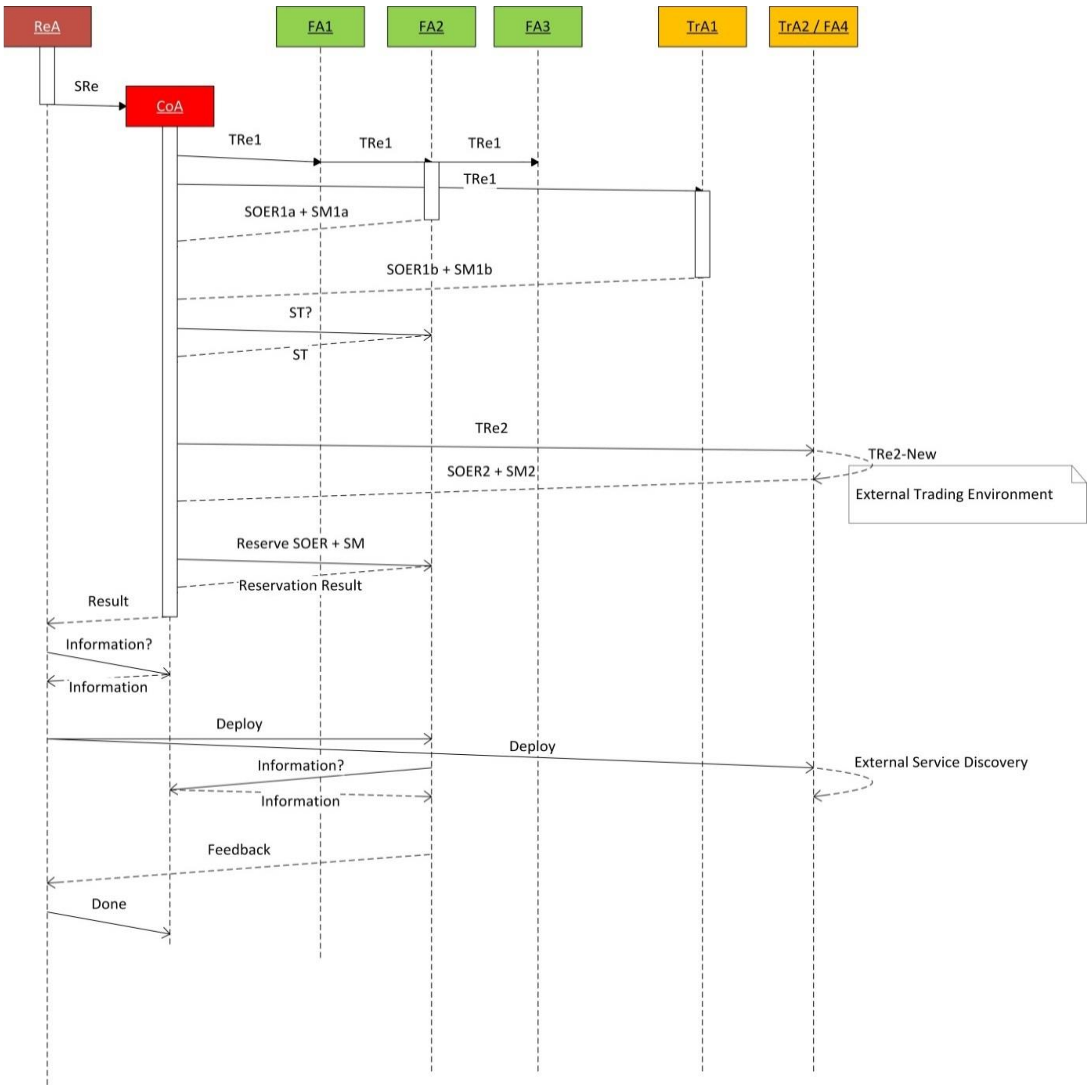


Fig. 35 - Sequence Diagram

The R-Model consists of two entities: the Service Request (SRe) and the Trading Request (TRe). The Composition Process, which builds a main part of the C-Model, starts with a **Service Request** (**SRe**, Definition 19). ACTAS assumes that the origin of the Service Request is in the application environment (cf. System Environment, chapter 8 and Fig. 35 - Sequence Diagram). The Request Agent (ReA), as a part of the application environment, creates with the SRe a Composition Agent (CoA) responsible for the Composition Process. During the Composition Process, a Trading Request (TRe) is used for the communication with the Trader Agents and/or Facility Agents, in order to find matching Service Offers, which are built with the entities of the S-Model.

In ACTAS, a Service Requester can request a service on behalf of several Service Clients. This becomes necessary among others for some Technical Services. For instance a communication service can only be performed with several Service Clients (cf. COR, Example 1). Therefore, a Service Request can contain several Client (Service) Requests (Client-Request in Definition 19).

However, the fact that a Service Request can be done on behalf of several Service Clients has consequences in comparison to a Service Description. A Service Description stands for one service. Its Service Modes describe alternatives of the Service Description of the same service. In a Service Request, each Client Request of a distinct Service Client holds information, which has to be dealt with in the Composition Process. Therefore, Client Requests are not alternatively, even when they are listed similar like the Service Modes of a Service Description. With its Common Part, the Service Request allows even the definition constraints and information, which are valid for the whole Composite Service. This is further discussed in the section 13.3, which covers the initialisation of the Composite Service Structure (CompSt) with the Client Requests and the Common Part of the Service Request. ´The case study in chapter 16 shows the application of General Characteristics in the Common Part of a Service Request, in order to check constraints of the composition of telecommunication features appearing in separate Client Requests.

The Client Requests use a special kind of Compatibility Characteristics; the so-called **Request Characteristics** (**RCh**) (cf. Definition 4). In the ontology of characteristics, the concept of an RCh has a specific relationship to concepts of a user ontology (in Fig. 28 - Principal ontological categorization of Semantic Characteristics: „Can_be_used_with“-relation), i.e. the semantic description (SemDescr in Definition 4 and Definition 5) of a Request Characteristic restricts the semantic context of Client Requests to specific user groups. For instance, the semantic description of the RCh "Reliability" in equation (22-20) of Table 34 - Request Characteristics (RCh) relates the Request Characteristic with a user group called "Administrator". A set of Request Characteristics of a Client Request build an interface of the Service Request held in a **Request Port** (a Request Port is a Service Port, which holds only Request Characteristics). ACTAS interprets a combination of Request Characteristics in a Request Port as an additional restriction of the addressed user group of the Service Request, since the addressed user group becomes the intersection of all user groups related with the individual Request Characteristic.

As discussed in section 10.3, an advertised Service Offer must also have a Request Port, in order to be "visible" for a Service Request. (A certain Option-Slot declares a Service Port, holding only references to Request Characteristics, as a Request Port (cf. section 10.2, Fig. 30, and Table 13 - Option-Slots of Service Ports).) In this way, ACTAS creates a clear distinction between an interface to Service Requests and an interface among only Service Offers. It is a support of **B2C** and **B2B** like interfaces inside of a Composite Service (cf. section 3.4). Since a B2C interface is always a server-client interface and addressing a certain costumer group. The Request Ports have direction Option-Slots (cf. Definition 10 and Table 13 - Option-Slots of Service Ports): the Request Port of a Service Request is designed as an OUT Port and the Request Port of a Service Offer as an IN Port.

The **Trading Request** (**TRe**) is a describing element, comparable to the "goal" in WSMO. In the idea of ACTAS, it is used as a multicast message, i.e. it is sent simultaneously to several Trader and Facility Agents, in order to find (principally) compatible Service Offers. For this purpose, the data structure given in this thesis has to be transformed into e.g. a XML formatted message. This leads to a general remark about all informal data structures of the S-Model and R-Model: they are done in a hierarchical format so that they easily can be translated into XML Schemas, in order to be transferred between the different agents. (In Fig. 35, the first Trading Request (TRe1) is forwarded by the Facility Agents, in order to achieve a kind of multicast). The agents will at least check the principal compatibility, i.e. they will test if they know a Service Offer with a Service Mode having an interface (Service Port) with the same set of Compatibility Characteristics as requested. This testing of principal compatibility (including the compatibility of the Option-Slots) was described in section 10.3 - Description of compatibility. Additionally, the agents could check the Merge Constraints (cf. Definition 16). A special Option-Slot in the Trading Request could demand such an additional test (cf. section 11.3).

In case of a successful compatibility testing, the agent will return the SOER and a reference to the found Service Mode (SM) (cf. Fig. 35). Thus, the CoA receives with the SOER the current state of the service. In order to build the Service Offer, the CoA might additionally ask the publishing FA for the Service Template (ST) (cf. Fig. 35). The SOER and the Service Mode could be used for a reservation of the Service Mode through the responsible FA on the base of the SOER (the Composite Service data structure (CompSt) in Definition 24 offers the field "Res-Info", which could be used in this direction). A reservation is useful for an improved resource management by the FA and a more reliable deployment of the service. The FA will publish an adapted SOER as a reaction on the reservation.

A **Trader Agent** can also react on a Trading Request. In Fig. 35, two Trader Agents are shown. The right one (TrA2) takes advantage of an external trading environment. In Example 5, such a scenario was already discussed in the context of the extended use of Semantic Characteristic. A Compatibility Characteristic could be introduced, which will be recognised by the Trader Agent, when its reference appears in the set of references of Compatibility Characteristics in the Trading Request. The Trader Agents can compose a new Composite

Service or act as a gateway to an external Trading Environment. In both case, they become a FA of this new and have to find an adequate Service Description with an appropriate set of references of Compatibility Characteristics.

Messages of the deployment are also in the principal Sequence Diagram of Fig. 35. After a positive result of the CoA, the ReA can have an information exchange with the CoA, in order to inform its application environment or to initiate the negotiation and grounding phase of the selected Facility Agents (cf. section 8.5 - Phase 5 – Grounding and Schedule of the Services).

Element	Description
Service Request (SRe) • Client Request • SRe-Common	Service Request (SRe) is the start of the Composition Process. It can contain several requests on behalf a Service Clients (Client Request). In a Common Part can be General Characteristics, which are valid for all Client Requests (SRe-Common).
Trading Request (TRe)	The Trading Request is comparable with the "goal" entity of WSMO, i.e. it holds the information about a requested service and is matched with the existing Service Offers by the agents.

Table 16 - Elements of the R-Model

11.1 Service Request (SRe)

Each Service Request (SRe) (cf. Definition 19) has a distinct identification SRe^{ID} and a reference to its Request Agent (ReA). The Common Part and each Client Request offer space for sets of references of General Characteristics. The Client Requests have identifications RM^{ID} introducing the labelling Request Mode (RM). A Request Mode holds a set of Request Ports (RP). A RP was earlier introduced as a Service Port holding only references to Request Characteristics as Compatibility Characteristics.

On a first glance, the data structures of a Service Request appear dual to the Common Part and the Service Mode of a Service Template (ST) (cf. Definition 8). This impression is supported by the fact that the Client Request as so-called Request Mode (RM) has at least one Request Port (RP), which holds the sets of references of Request Characteristics. However, this impression is misleading. In a Service Description the Service Modes are alternative descriptions of the same service and the Common Part adds common information for the Service Modes. In a Service Request, each Client Request is an own standing piece of information, which will be used in the Composition Process. It has its individual reference to an agent, the assumedly existing Personal Agent (PA), whereas the whole ST belongs to only one Facility Agent.

The Common Part of a Service Request is not keeping information, which is an extension of the information of each Client Request, as one could assume in comparison to a Service Description. The General Characteristics referenced in the Common Part of a Service Request become information for the Common Part of the Composite Service. This is further described in section 13.3.

Example 16 **Alternative Client Requests**

In ACTAS, a Service Request can be done for several (potential) Service Clients. For every Service Client, several services can be requested like for instance when a customer wants to book a flight, reserve a hotel, and hire a car. However, how shall these services be delivered? Must it be one travel agency as Service Provider or are several Service Providers allowed or even wished? Is the Service Client also happy with only a subset of these services?

Additionally, the Client Request contains a reference to an Actor Service Template (AST). It is a data structure, which becomes relevant in a future version of ACTAS, when feedback based learning of users' preferences will be added. Currently, in the C-Model, each Client Request becomes an Actor Service Offer (ASO), which is a dual data structure to a SO and the initialisation of the composite data structure of the Composition Process. The AST could become the base of the ASO. The Client Request would adapt the RM of the ASO.

Definition 19. Service Request (SRe)

$$\begin{array}{lcl} \text{SRe} & \equiv & (\text{SRe}^{\text{ID}}, \text{ReA}^{\text{Ref}}, \text{SRe-Common}, \text{Client-Request}^{\text{Set+}}) \\ \text{SRe-Common} & \equiv & ((\text{GCh}^{\text{Ref}})^{\text{Set}}, \text{SRe-Env}) \\ \text{Client-Request} & \equiv & (\text{RM}^{\text{ID}}, \text{PA}^{\text{Ref}}, AST^{Ref}, \text{RP}^{\text{Set+}}, (\text{GCh}^{\text{Ref}})^{\text{Set}}, \text{SRe-Env}) \\ \text{RP} & \equiv & (\text{RP}^{\text{ID}}, (\text{RCh}^{Ref})^{\text{Set}}, \text{SRe-Env}) \end{array}$$

Service Request (SRe) consists of

- Name of Service Request (SRe^{ID})
- Reference of the responsible Request Agent (ReA^{Ref})
- A Common Part of the Service Request (SRe-Common)
- A set of Client-Requests (at least one element) ($\text{Client-Request}^{\text{Set+}}$)
-

A Client-Request (a Request Mode (RM)) contains the following information:

- Name of Request Mode (RM^{ID})
- Reference of possibly existing Personal Agent (PA^{Ref})
- Information of Service Client (Actor Service Template (AST))
- Set of Request Ports (at least one element) ($\text{RP}^{\text{Set+}}$)
- Set of References of General Characteristics ($(\text{GCh}^{\text{Ref}})^{\text{Set}}$)
- Set of References of Request Characteristics ($(\text{RCh}^{\text{Ref}})^{\text{Set}}$)

$\text{SRe-Env} \equiv (\text{Va-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}}, \text{SRe-Option-Slot}^{\text{Set}})$

- Set of Value Constraints ($\text{Va-Co}^{\text{Set}}$)
- Set of Exchange Constraints($\text{Ex-Co}^{\text{Set}}$)
- Set of Option-Slots for SRe ($\text{SRe-Option-Slot}^{\text{Set}}$)

In Example 16, **Alternative Client Requests** are discussed. In other Service Discovery approaches, several requests have to be posed or a booking service must be applied. In ACTAS, the Service Request could have directly Alternative Client Requests: These are Client Requests done for the same (potential) Service Client, i.e. they contain a reference to the same Personal Agent (PA). In this way, the duality to the ST becomes bigger, because the alternative Client Requests equally belong only to one agent.

11.2 Trading Request (TRe)

Definition 20. Trading Request (TRe)

TRe ≡ ($\mathrm{TRe}^{\mathrm{ID}}$, $\mathrm{Agent}^{\mathrm{Ref}}$, $(\mathrm{CCh}^{\mathrm{Ref}})^{\mathrm{Set}}$, TRe-Env)

Service Request (TRe) consists of

- Name of Service Request ($\mathrm{TRe}^{\mathrm{ID}}$)
- Reference to the sending Agent ($Agent^{Ref}$)
- Set of references to the Compatibility Characteristics, $(\mathrm{CCh}^{Ref})^{\mathrm{Set}}$
- Environment description (TRe-Env)

TRe-Env ≡ ($\text{Va-Co}^{\mathrm{Set}}$, $\text{Ex-Co}^{\mathrm{Set}}$, $\text{TRe-Option-Slot}^{\mathrm{Set}}$)

- Set of Value Constraints ($\text{Va-Co}^{\mathrm{Set}}$)
- Set of Exchange Constraints($\text{Ex-Co}^{\mathrm{Set}}$)
- Set of Option-Slots for TRe ($\text{TRe-Option-Slot}^{\mathrm{Set}}$)

In comparison to the SRe, the Trading Request (TRe) is a simpler informal data structure. Besides a distinct identification ($\mathrm{TRe}^{\mathrm{ID}}$), which will be quoted in the response of the Facility Agent or the Trader Agent respectively (cf. Fig. 35), it contains a reference to the posing agent (in Definition 20 the $Agent^{Ref}$ will be mostly a reference to the Composition Agent, CoA). Important are the references of the Composition Characteristics $(\mathrm{CCh}^{Ref})^{\mathrm{Set}}$ (this includes references to Request Characteristics as a special kind of Composition Characteristics), since they allow the Trader Agents and the Facility Agents to determine the principal compatibility. The environment specifications (TRe-Env) allow also the application of Value Constraints and Merge Constraints. Even Exchange Constraints concerning the Char Properties of the reference Compatibility Characteristics could be performed for extended tests by the agents.

11.3 The environment description of the R-Model

Since the R-Model is an extension of the informal S-Model the constraints can be posed in the same way as discussed in the section 10.4 - Constraints in the Service Description. Some possible Option-Slots for the Request Ports of the Client Request and the Trading Request (TRe) are listed in Table 17. The “overwrite” Option-Slot makes only sense for SOERs. The “test” Option-

Slot allows the control of the extent of the compatibility test. The addressed Facility Agents and/or Trading Agents can be selected.

In the evaluation, the use of General Characteristics in the Common Part of the Service Request (SRe-Common) as "building blocks" is covered in Case Study 1: Technical Services with translation. The case study uses an adaptation of the "ExchangeProperties" term of an Exchange Constraint given in the environment description of the Semantic Characteristic (Char-Env in Definition 4) through an "exchangeProperties" Option-Slot (cf. Table 14) in the environment description of SRe-Common (SRe-Env in Definition 19). The necessary references of the addressed Service Properties were already considered with Definition 15.

Option-Slots	Semantic and Attributes
Direction	The Request Ports and TRe are normally OUT Ports, since they look as clients for a service. However, the TRe is also use for non-directed composition • *direction*(OUT) – Client Port, "OUT Port" • *direction*(IN) – Server Port, "IN Port"

test (PRINCIPAL)
test(MERGE)
test(FULL)
	Normally the principal compatibility is tested ("test" Option-Slot is not set or "test(PRINCIPAL)")
	• test(MERGE) – Value Constraints and Merge Constraints are checked
	• test(FULL) – Additionally eventually existing Exchange Constraints are tested

Request	This Option-Slot is standard for the Request Ports of the Client Requests i.e. all Compatibility Characteristics are Request Characteristics The TRe reflects the Option-Slot of the Open Port, that means of the Service Port, for which the Composition Process tries to find a compatible one
facility-agents(list-of-FAs) trader-agents(list-of-TrAs)	Enumeration of agents, which should be preferentially asked for compatible Service Offers in the composition or Trading Process. In a closer specification, it could be expressed if these agents are compulsive.

Table 17 - Option-Slots of Request Ports/TRe

ACTAS - COMPOSITION MODEL (C-MODEL)

12 Introduction to Composition Model (C-Model)

The Service Model (S-Model) (chapter 9) describes the services from the perspective of a Service Provider and distinguishes between Service Templates (ST) (cf. Definition 8) and Service Offer Export Records (SOER) (cf. Definition 9) for the Service Description. These data structures are used for the advertising of Service Offers by the Facility Agents. The data structures of Service Offers exist only in the declarative environment of the agents. ST and SOER entities are applied for their building. They are also used by the Trader Agents and the Composition Agents during the trading and Composition Process.

The Request Model (R-Model) (chapter 11) changes the describing perspective to the Service Requester and the Service Clients. It offers the **Service Request** (**SRe**) data structure for the Request Agent (ReA) of the application environment, in order to pose a request for a service and to start the Composition Process. The Composition Process is performed by a specifically created Composition Agent (CoA), whose policies and algorithms can be adapted to the needs of the application environment, since the creation is done through the ReA.

The SRe is the base for the building of Actor Service Offers (ASO) in the declarative environment. The CoA involves the Facility Agents and the Trader Agents for the discovery and matching of Component Services. For this purpose, the **Trading Request** (**TRe**, Definition 20), the second entity of the R-Model, is applied. The TRe is also used by the Trader Agent for its trading. The Trading Process is comparable with the Composition Process, but in opposite to the Composition Process, the Trading Process of the Trader Agents could be only based on the Service Templates, in order to compose services, which are principally possible. The Trader Agents can collect service candidates fitting their policies or compose new Composite Services. In the latter case, they will also act as a Facility Agent of this new Composite Service. Another case, when a Trader Agent would act as a Facility Agent, is given, when the Trader Agent acts as a gateway to an external trading environment (cf. Fig. 35 - Sequence Diagram). However, the description of the C-Model will concentrate on the Composition Process.

The main difference between ASO and SO is the point of view of their descriptions, due to their origin: a SO takes the describing perspective of a Service Provider (it contains a references to the ST, SOER, and the FA), whereas the ASO is a description from the perspective of a Service Requester/Client (it contains reference to the SRe, Client Request, and the PA). Nevertheless, a duality between Service Description and Service Request can be seen in the roles of the Facility Agent and the Personal Agent. The eventually existing Personal Agent (PA) of a Service Client can be perceived as the “Facility Agent” of the Client Request.

In the future research, this duality becomes more evident, when learnt information based on feedback and users' preferences will be stored in an **Actor Service Template** (**AST**) by the PA. The AST could be useful for the default information of the ASO, especially Option-Slots, like the one referencing preferred FAs, could control the upcoming Composition Process. (The definition of the Service Request contained already a reference to a user specific AST in every Client Request, cf. Definition 19.)

The C-Model is based on a declarative environment like it can be implemented with SICStus Prolog (cf. Evaluation, chapter 14). Such an environment eases the testing and backtracking, in order to achieve a wished Autonomic Service Discovery. However, the methods of the Property Classes have to be integrated, which establish the framework character of ACTAS, realise the constraints and allow an adaptation of the Service Description (cf. S-Model). It is a challenge to integrate possibly imperative programmed, external entities into a declarative environment, especially when there is no direct control over their values and behaviour. Therefore, ACTAS uses the concept of objects inside of the declarative environment. These objects also allow the encapsulation of the access of the external methods. In the declarative environment only two predicates are used: op/5 and test/4. The former applies the constraints resulting of the call of the methods to clones of the currently valid objects. This cloning is necessary for the support of the backtracking and the monotony of the constraints. The test/4 predicate does not make a new object available, i.e. it realises a testing on the current objects, but the constraints are not internally saved. Therefore there will be not guarantee of monotony of constraints, when test/4 is applied. The description of the Composition Process in the next chapter will have a closer look at this matter. It also describes the steps of the Composition Process in a general way. In the end, it depends on the policies of the Composition Agent (CoA) and the interpretation of the Option-Slots in the Service Description and the Service Request, in order to determine how the steps of the Composition Process are performed as well as how the discovery and composition is done. Therefore, alternative approaches are discussed in an extra chapter and the Evaluation chapter.

Element	Description
Composite Structure (CompSt) • **CompSt-plus** • **Open Port**	The goal of the Composition Process is a **Composite Structure** (**CompSt**) without any **Open Port** that satisfies the Service Request, i.e. all Service Ports of the CompSt are composed with a compatible Service Port including all Request Ports of the Service Request. Normally, this also means that Service Properties of the compatible Service Ports are "merged" (Merge Constraints were applied successfully) and the Exchange Constraints are fulfilled. The CompSt-plus data-structure is used during the Composition Process. It offers additional space for the constraints and Open Ports.

Element	Description
(Declarative) Objects • **Char Property Object** • **Merge Property Object** • **Exchange Property Object**	Objects are introduced in the declarative environment as handles for the access of the implementation of Property Classes (cf. Definition 6 in section 9.2). Since there are three different kinds of Property Classes, three different kinds of Property Objects are handled. The methods of the Property Classes, used in constraints, are translated into op/5, or test/4 predicates, respectively. A Char Property Object keeps the values and methods of a Service Property (due to the fact that every Service Property was once declared as a Char Property in the context of a Semantic Characteristic (cf. S-Model Definition 4)). The methods of a Char Property Class are used for Value Constraints (cf. Definition 14), (Exchange Constraints (cf. Definition 18)), and getting information about the values of a Service Properties (e.g. “printValues”, cf. section 9.2). A Merge Property Object is created with the application of the Merge Constraint, which “merges” two Service Properties with the same Char Property Class. In a directed composition, such a “merge” can be seen from three perspectives: (1) the client side (what is requested?), (2) the server side (what is offered?), and (3) the merged side (what is a compromise?). Therefore, the “merge” constructor used for the Merge Constraint can lead to new values in Merge Property Object. The Merge Property Class must offer Import and Export methods, in order to deal with the right view in the Exchange Constraints. An Exchange Property Object just enables the use of methods, which correlates several Service Properties. These methods are used in Exchange Constraints.
Service Offer (SO)	The agents use the descriptions of the Service Template and the Service Offer Export Record (SOER), in order to build a Service Offer (SO). The data structure of SO is comparable to the one of ST, but it includes the (declarative) objects and references of the Service Properties to their current objects.
Actor Service Offer (ASO) • **Actor** • **Actor Service Template (AST)**	Currently, only the Service Client is called an “actor”. The origin idea of an “actor” in ACTAS is an agent, who can potentially supply feedback for learning. An Actor Service Offer (ASO) is constructed from each Client Request of a Service Request (SRe) (cf. Definition 19). An ASO is a dual data-structure to the SO. The resulting ASO(s) are the initialisation of the Composite Structure (CompSt-plus) used for the Composition Process. In a future version of ACTAS, the feedbacks of the Service Clients will be used for the learning of the preferences save in an Actor Service Template (AST). The AST will help to construct the ASO or can be used for the management of service offered privately by the Personal Agent. Thus, the Service Client could keep information about his payment (service) preferences or the currently accessible communication facilities.

Table 18 - Elements of the C-Model

13 C-Model: The Composition Process

The Composition Model (C-Model) covers three phases of Service Discovery in Fig. 2 - Phases of the extended life cycle of e-service: Trading (phase 2), Matching (phase 3), and Checking of constraints (phase 4). The goal of the C-Model is to find a Composite Structure (CompSt in Definition 24), which fulfils the Service Request (SRe in Definition 19) and its resulting constraints. In general, the C-Model of ACTAS works in six steps:

1. Getting the information of the S-Model and R-Model
2. Initialization of the extended Composite Structure (CompSt-plus, Definition 25)
3. Discovering and composing of principally compatible services.
4. Checking of Merge Constraints
5. Checking the Exchange Constraints (Ex-Co)
6. Post-Processing: Checking of other constraints like the availability of resources.

Table 19 - Steps of the Composition Process

The specifications of (Semantic) Web Services are based on informal data structures independent from the deployed services as proposed by Berners-Lee [Ber2003]. Solutions of (Autonomic) Service-Oriented Computing put on these data-structures. Nevertheless, the need for the inclusion of the reference of active elements was recognized. For instance, the reference of a WSML description of a mediator in WSMO (cf. [StGrAb2007 section(s) 287–311]). The Property Classes (cf. section 9.2) take on this idea with the grounding descriptions (GroundingDescr in Definition 6). In the declarative environment of the C-Model, the access of the algorithms of the Property Classes is realised and wrapped through objects. In this sense, the Service Properties gained an "active" behavioural semantic. Thus, the declarative data structures of the C-Model (e.g. SO and ASO) are not simply informal like the ones introduced in the S-Model and the R-Model. They need the execution environment of the agents.

The entities of the S-Model are used for the creation of **Service Offers** (**SO**, cf. Definition 22). In a dual way, each Client Request of a Service Request (SRe) becomes a so-called Actor Service Offer (ASO, cf. Definition 23) (Step 1 in Table 19 - Steps of the Composition Process). The created ASOs become the initialisation of the data-structure of the Composite Service. It is the goal of the Composition Process to deliver complete Composite Services following the specific policies of its CoA. This can also mean that the CoA collects a set of Candidate Services instead of a single service. The Composition Process forms the main part of the Composition Model (C-Model).

In a successful case, the Request Agent can ask for information of the Composition Agent (cf. Fig. 35 - Sequence Diagram). For the return of a whole Composite Structure the values of the

```
:- use_module(library(objects)).

:- class point =
[
public x:float = 1.0,
public y:float = 2.0
private handle
].

Self <- create(X, Y, GDescr) :-
Self << x(X),
Self << y(Y),
getHandle(GDescr, Handle),
Self << handle(Handle).

Self <- callMethod(Method, PList,
    OList, OK) :-
Self >> handle(Handle),
callWithHandle(Handle, Method, PList,
    OList, OK).
Self <- printValues(Stream) :-
Self >> x(X),
Self >> y(Y),
format(Stream, '(~w, ~w)', [X,Y]).

Self <- clone(PointObj_clone) :-
create(point, PointObj_clone),
Self >> x(X),
PointObj_clone << x(X),
Self >> y(Y),
PointObj_clone << y(Y),
Self >> handle(Handle),
getNewHandle(Handle, NewHandle),
PointObj_clone << handle(NewHandle).

:- end_class point.
```

Code 1 - Object in SICStus Prolog with idea of handle

Service Properties should be documented. For this purpose, the method "printValues" was suggested for the Char Property Classes in section 9.2 - S-Model: Property Classes (cf. also Code 1 - Object in SICStus Prolog with idea of handle).

The explanations of this chapter are often done through rules in order to describe the declarative environment. A rule comprises a premise and an entailment consisting of expressions about objects or a statement about concrete actions like for instance the call of another rule.

13.1 The Property Objects/Classes in the C-Model

The Property Objects of the declarative environment of ACTAS only wrap the access of the implementation instances of the properties. Several instances of the implementation of the methods of a Property Class can be external. The description of grounding of Property Classes (*GroundingDescr* in Definition 6) contains more information. In Code 1, the idea of a handle, held by an object, is schematically shown. It shows a point object, realised with the SICStus Prolog library extension "objects". It has three attributes (in Prolog called "slots"): the Cartesian coordinates of the point and a handle, which assumedly gives access to an implementation instance of a Property Class. The constructor of the object takes the Grounding Description of the Property Class and calls the predicate getHandle/2, in order to receive such a handle. A method call needs this handle. The Grounding Description must give a hint, which adaptations in the Prolog object are necessary, in order to achieve a wished transparency of the access of a method. The object list (OList) in this call has to be adapted, in order to be processable by the implementation instance (in Code 3 - op/5 and test/4, showing a realisation of the predicates op/5 and test/4, the adaptation is assumedly done by the predicate adaptObjects/5 in one direction and readaptObjects/5 in the opposite direction). The methods themselves must return a BOOLEAN (in the code example called "OK"), in order to integrate them into the declarative environment.

```
<WebMethod(EnableSession:=True)> Public Sub SetName(ByVal CurrentName As String)
    Dim t As New Test
    t.setName(CurrentName)
    Session.Add("Name", t)
End Function

<WebMethod(EnableSession:=True)> Public Function GetName() As String
    Dim t As Test
    t = CType(Session("Name"), Test)
    Return (t.getName)
End Function
```

Code 2 - Statefull Web Service Methods in Visual Basic

The handle for the access of the implementation instance of a Property Class could be similar to the one realised through the WSML description of a mediator in WSMO (cf. [StGrAb2007 section(s) 287–311]), which addresses a Web Service implementing the mediation; i.e. the Grounding Description could contain the URL of a WSDL description of a Web Service.

An alternative approach to Web Services would be the supply of a Prolog module for the implementation of the methods of the Property Class. This module could be saved in the ontological repository of the Property Classes. Such a Prolog module can directly be integrated into the object concept of Prolog, in order to have separated data for each object instance. Extending this idea, even the objects giving access to Web Services or other external implementation instances could be wrapped as modules and stored in the ontological repository of the Property Classes. In this way, the access and implementation of the Property Classes could become transparent for ACTAS, secure and highly adaptive. This is future research, the creation and management of this handle goes beyond the scope of this dissertation. In the following thesis, it will be spoken about the implementation of Property Classes and its instances accessed through Property Objects.

Every new Property Object created in the declarative environment should access the implementation through its own instance and context. This means for example for a Web Service that it must be statefull for each of these contexts, in order to keep hold of the (internal) constraints and attributes. This will be important, when the Property Object holds information, in order to achieve monotony of the constraints. Otherwise, a Property Object, which handles the access of a stateless Web Service, must be programmed in such a way that it manages the settings. Such kind of Property Object will have to send in a message to the Web Service the complete list of settings each time, when it accesses the Web Service, in order to test their fulfilment. This is not really feasible. Although a Web Service is not statefully in its original idea, it can be extended in this direction. Code 2 - Statefull Web Service Methods in Visual Basic – shows such an extension: so-called sessions can be used, in order to keep hold of a context (here the information of a class "test").

Service Properties, which are defined as Char Properties with Char Property Classes, and possibly Merge Properties, which are the results of Merge Constraints described with Merge Property Classes, are used for the keeping of information in ACTAS, i.e. they encapsulate attributes and implement transparent constraints on these attributes, and in ACTAS these constraints are called **Attribute Constraint**s. Additionally, these properties keep internally further constraints, resulting from the applications of methods. Therefore, at least Web Services implementing Char Property Objects have to be statefull.

Exchange Property Objects are used for the access of the methods of the Exchange Property Classes, in order to correlate several Service/Char Properties possibly declared with different Char Property Classes. In a method of an Exchange Property Object, it often does not make sense to create a common entity for the storing of information. However, Exchange Constraints use the information of the Property Objects given in the call of the method, in order to determine the fulfilment of their common constraints. Thus, they take at least advantage of the statefullness of the other Property Object. Since an Exchange Constraint is built of several method calls (cf. Definition 18), it is nearly unimaginable to think about Exchange Property Object as stateless.

Definition 21. Property Object and its Predicates in C-Model

A Property Object wraps the access of a Property Class in the C-Model. It has the following general features:

- A behavioural semantic with its methods
- A Knowledge Base (KB in rule (ExOp))
- Application Ontologies ($\mathfrak{O}$ in rule (ExOp))
- Internal, unchangeable Attribute Constraints (**logAttrCo** in rule (ExOp))
- Internal Value Constraints (resulting of the method applications) ($\mathbf{logValueCo}^{\mathbf{Set}}$ in rule (intOp))

The methods of the Property Class are implemented by instances handled by Property Objects. The Property Objects are accessed through two predicates

- Predicate op/5 for a monotone collection of constraints: $op(PropertyObject, Method, Para^{Set}, InputO^{Set}, OutputO^{Set})$
- Predicate test/4 for testing of constraints at a time point: $\boldsymbol{test(PropertyObject, Method, Para^{Set}, Object^{Set})}$

In order to unify the view on a Property Object in the declarative environment of the C-Model, a Property Object is defined as having a behavioural semantic through its methods, a knowledge base, and its application ontologies (cf. Definition 21). A Property Object encapsulates constraints for the values of its attributes, the so-called **Attribute Constraints**. Attribute Constraints are given with the definition of the Property Classes and their semantic. They are transparent and valid for every property declared with the Property Class (cf. S-Model Definition 4). For instance, the implicit plausibility check of the OWL-S capability sketched in Example 7 is an implementation of Attribute Constraints. The behaviour and the attributes of the Property Objects are out of control of ACTAS. However, the methods and their access are transferred into predicates for the objects, in order to tackle their integration into the declarative environment of the C-Model of ACTAS.

These predicates are op/5 and test/4 (cf. Definition 21). In the declarative environment, these predicates are used for the evaluation of the constraints. As discussed earlier, the implementation of Property Objects should be firstly realized statefully, i.e. the state after the last application of op/5 should be kept and should still be accessible. Secondly, the information before an application should be kept, in order to allow a backtracking. For this purpose, the Property Object is cloned before the application. The cloning means for the implementation instance the creation of a new handle and the taking over of the state. This is schematically shown in Code 1 - Object in SICStus Prolog with idea of handle.

The internal constraints of Property Objects create a wished monotony. The integration of objects into a declarative environment is a challenge, since they encapsulate information and are rather imperative. Therefore, their integration and the dealing with internal constraints are discussed in general in the rest of this section with the two rules (ExOp) and (intOp).

Schematically, the rule (ExOp) looks at the call of a method through the predicate op/5 as a declarative call with sets of input objects, output objects, and parameters. The checking of constraints can be seen as the processing of a list of operation calls (Co^{List}). Each operation call adds new internal Value Constraints to the attributes of the involved objects (the rule (intOp) takes this internal point of view for one involved object). Thus the list of constraints ($\mathrm{Co}^{\mathrm{List}}$) can be interpreted as a conjunction of constraint setting operations.

$$(\mathrm{ExOp})\,\frac{\mathrm{O}^{\mathrm{Set}},\,\mathrm{op}(\mathrm{MethodO},\,\mathrm{Method},\mathrm{Para}^{\mathrm{Set}},\,\mathrm{InputO}^{\mathrm{Set}},\,\mathrm{OutputO}^{\mathrm{Set}}) \wedge \mathrm{Co}'^{\mathrm{List}}}{\mathrm{O}'^{\mathrm{Set}},\,\mathrm{Co}'^{\mathrm{List}}}\quad \text{if}$$

$$\begin{gathered}\mathrm{OutputO}^{\mathrm{Set}} \text{ are the cloned objects of } \mathrm{InputO}^{\mathrm{Set}},\\ \text{The application of Method} \in \mathrm{Method}^{\mathrm{Set}} \text{of PC of MethodO (cf. Definition 6)}\\ \text{was successful with } \mathrm{Para}^{\mathrm{Set}} \text{ and } \mathrm{ad}(\mathrm{OutputO}^{\mathrm{Set}}),\\ \mathrm{O}'^{\mathrm{Set}} \leftarrow (\mathrm{O}^{\mathrm{Set}} \cup \mathrm{OutputO}^{\mathrm{Set}})/\mathrm{InputO}^{\mathrm{Set}}\end{gathered}$$

The declarative environment handles the backtracking and keeps the current set of all Property Objects in $\mathrm{O}^{\mathrm{Set}}$. A subset of $\mathrm{O}^{\mathrm{Set}}$ is used as a set of input objects ($\mathrm{InputO}^{\mathrm{Set}} \subseteq \mathrm{O}^{\mathrm{Set}}$) of op/5. In a successful case, i.e. the method call could be applied to cloned objects of the input objects (cf. rule (ExOp)); the operation op/5 will produce a set of new output objects ($OutputO^{Set}$). Finally, ACTAS will build a new set of current Property Objects ($\mathrm{O}'^{\mathrm{Set}}$) through the integration of the output objects.

The rule (ExTest) as an alternative rule to rule (ExOp) does not produce a new set of Property Objects. The O^{Set} stays unchanged. The input objects $InputO^{Set}$ are only cloned to $ClonedO^{Set}$, in order to use them for the application of the method. Thus, the constraints are only tested, but not preserved. In the rules, the $\mathrm{Method}^{\mathrm{Set}}$ of the Property Class (PC) (cf. Definition 6) is mentioned. The object MethodO realises a handle to an implementation instance of this PC. The possibly adaptations of the objects used in the method call, also mentioned in the conditions of the rules with the function ad/1, was discussed above in context of Code 1.

$$(\mathrm{ExTest})\ \frac{\mathrm{O}^{\mathrm{Set}}, \mathrm{test}(\mathrm{MethodO}, \mathrm{Method}, \mathrm{Para}^{\mathrm{Set}}, \mathrm{InputO}^{\mathrm{Set}}) \wedge \mathrm{Co}'^{\mathrm{List}}}{\mathrm{O}^{\mathrm{Set}}, \mathrm{Co}'^{\mathrm{List}}}\ \text{if}$$

$$\text{ClonedO}^{\text{Set}} \text{ are the cloned objects of } \text{InputO}^{\text{Set}},$$
$$\text{The application of method} \in \text{Method}^{\text{Set}} \text{of PC of MethodO (cf. Definition 6)}$$
$$\text{was successful with } \text{Para}^{\text{Set}} \text{ and } \text{ad}(\text{ClonedO}^{\text{Set}})$$

The creation of new objects and the incorporation of constraints is necessary, in order to reassemble a stack of information for the backtracking process. Property Objects cannot be directly included in the declarative environment due to their encapsulation of data and likely imperative character. If no new solution for the current output objects of the op/5 operation can be found, the backtracking will go back to the earlier called op/5 and its stacked objects. In Code 3 - op/5 and test/4, a possible approach for the implementation of the predicates op/5 and test/4 is shown; the latter predicate does not make the output list with the new objects available, but has also to test the constraints with cloned objects, in order not to destroy the original information. In this way, the validity of the constraints is only tested for the time point. (Remark: In the code, the parameters show with the symbols "+" and "-", which are necessary input ones and which ones must be a variable for output. (cf. [Car2009])).

```
op(+PropertyAccess,+Method,+ParameterList,+ObjectInList, -ObjectOutList) :-
     cloneObjects(ObjectInList, ClonedObjectList),
     adaptObjects(PropertyAccess, Method, ParameterList, ClonedObjectList,
               MethodObjectList),
     callMethod(PropertyAccess, Method, ParameterList, MethodObjectList),
     readaptObjects(PropertyAccess, Method, ParameterList, MethodObjectList,
               ObjectOutList).

test(+PropertyAccess, +Method, +ParameterList, +ObjectList) :-
     cloneObjects(ObjectInList, ClonedObjectList),
     adaptObjects(PropertyAccess, Method, ParameterList, ClonedObjectList,
               MethodObjectList),
     callMethod(PropertyAccess, Method, ParameterList, MethodObjectList).
```

Code 3 - op/5 and test/4

The set of parameters $(Para^{Set})$, another input of the introduced operation op/5, can be used for the initialization or setting of the attribute values through Value Constraints. They can be also used for the control of the behaviour of the methods of an Exchange Constraint (cf. Example 15). In Example 7, it was sketched how a Char Property holding OWL-S Service Descriptions could be initialized through a method with a parameter giving an URL pointing to a concrete OWL-S description. Similar, a Char Property Object dealing with WSML descriptions of WSMO could have a method with some parameters allowing the initialisation of the object alternatively with a WSML service, goal, capability, interface, or mediator description data preferably accessed through a URL.

$$(\mathrm{intOp})\,\frac{\mathfrak{O},\,\mathrm{KB},\,\mathrm{logAttrCo}^{\mathrm{Set}},\,\mathrm{logValueCo}^{\mathrm{Set}}}{\mathfrak{O}',\,\mathrm{KB},\,\mathrm{logAttrCo}^{\mathrm{Set}},\,\mathrm{logValueCo}'^{\mathrm{Set}},\,\Theta=\{\theta|\mathfrak{O}'\cup\mathrm{KB}\vDash\theta(\mathrm{logCo}^{\mathrm{Set}})\}}\ \text{if}$$

$$\mathrm{logCo}^{\mathrm{Set}}\leftarrow\mathrm{logAttrCo}^{\mathrm{Set}}\cup\mathrm{logValueCo}'^{\mathrm{Set}},$$
$$\mathrm{logValueCo}'^{\mathrm{Set}}\leftarrow\mathrm{logValueCo}^{\mathrm{Set}}\cup\mathrm{logValueOpCo}^{\mathrm{Set}}$$
$$\mathfrak{O},\,\mathfrak{O}',\,\mathrm{KB}\text{ belong to Object of applied method}\in\mathrm{Method}^{\mathrm{Set}}$$

The rule (ExOp) illustrated an external view on the method call. Since the Property Object only wraps the method access, it has to be clarified how the successful application of op/5 is implemented internally. Therefore, the internal view of one of the involved objects is added through rule (intOp). The rule describes only schematically the effects of a method application in a declarative way. In the end, the internal behaviour of a method is out of control of ACTAS, but one can generally demand that the new input is tested against the existing knowledge and the given semantic context in a monotone way. The demand of monotony and the creation of new objects satisfying the new internal constraints are important for a reliable backtracking. In rule (intOp), a knowledge base KB represents the knowledge generally given with the attributes and the behavioural semantic of the object. A set of ontologies $\mathfrak{O}$ symbolizes a specific knowledge of the current semantic context. The Property Class might have methods for the setting of ontologies. SWS Service Descriptions like WSMO support ontologies and their mediation directly. For example a WSML description can contain references to ontologies. A mapping of these references to concrete ontologies allows an extension of the ontology set $\mathfrak{O}$. The earlier discussed Attribute Constraints are represented through internal logical constraints named $\mathrm{logAttrCo}^{Set}$. Finally, the internal logical Value Constraints $logValueCo^{Set}$ should have been collected through earlier applications of methods of the Property Object. The collection of logical Value Constraints is important for achieving the intended monotone character of the method applications.

The application of a method $\in Method^{Set}$ in rule (intOp) leads to a change of the known ontologies ($\mathfrak{O}'$) and/or an extension of the set of logical Value Constraints (set $\mathrm{logValueCo}'^{Set}$). The concrete extension $\mathrm{logValueOpCo}^{Set}$ is an internal result of the method application. However, these changes are only accepted, when the new logical constraints ($\mathrm{logCo}^{Set} \leftarrow \mathrm{logAttrCo}^{Set} \cup \mathrm{logValueCo}'^{Set}$) can be entailed from the knowledge base and the currently valid ontologies $\mathfrak{O}' \cup \mathrm{KB} \vDash \theta(\mathrm{logCo}^{Set})$. The entailment contains the application of a substitution ($\theta\colon \mathrm{var}(\mathrm{logCo}^{Set}) \rightarrow ID$), since the constraints can contain logical variables ($\mathrm{var}(\mathrm{logCo}^{Set})$).

The application of op/5 in rule (intOp) is only called successful, when the entailment leads to a possible substitution, a logical model. A set of likely several models is the result: model set $\Theta = \{\theta|\mathfrak{O}' \cup \mathrm{KB} \ \vDash \theta(\mathrm{logCo}^{Set})\}$. The consideration of logical variables and the likely existence of several logical models are due to the ambiguity of the values of the attributes, and should be incorporated in the backtracking process. This is done in the second approach of an implementation of the predicate op/5 and test/4 in Code 4 - Getting Variants with op/5 and test/4. The predicate get_variant/7 is added, which produces a parameter for the method call (line 16) of the Property Object. In this approach, it is assumed that the methods of the Property Classes are able to return another variant, when explicitly demanded. The number of possibly asked variants is restricted with MaxVariant (line 2). Again the predicate test/4 does not return the new object list (lines 5-9). The predicate go_ahead_with_variant/6 accepts the new object list at first. However, when the testing with this variant of the output objects comes to a retry, it calls get_variant/7 again for the production of a new variant.

```
1.   op(PropertyAccess,Method,ParameterList,ObjectInList, ObjectOutList) :-
2.        Variant is 1, MaxVariant is 5,
3.        get_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList,
4.                  ObjectInList, ObjectOutList).

5.   test(PropertyAccess, Method, ParameterList, ObjectList) :-
6.        Variant is 1, MaxVariant is 5,
7.        get_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList,
8.                  ObjectInList, ObjectOutList),
9.        destroy_objects(ObjectOutList).

10.  get_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList, ObjectInList,
11.                 ObjectOutList) :-
12.       Variant =< MaxVariant,
13.       cloneObjects(ObjectInList, ClonedObjectList),
14.       adaptObjects(PropertyAccess, Method, ParameterList, ClonedObjectList,
15.                 MethodObjectList),
16.       callMethod(Variant, PropertyAccess, Method, ParameterList, MethodObjectList),
17.       readaptObjects(PropertyAccess, Method, ParameterList, MethodObjectList,
18.                 ObjectCandList), !,
19.       go_ahead_with_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList,
20.                 ObjectInList, ObjectCandList, ObjectOutList).

21.  go_ahead_with_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList,
22.                 ObjectInList, ObjectCandList, ObjectOutList) :-
23.       ObjectOutList = ObjectCandList.

24.  go_ahead_with_variant(Variant, MaxVariant, PropertyAccess, Method, ParameterList,
25.                 ObjectInList, ObjectCandList, ObjectOutList) :-
26.       destroy_objects(ObjectCandList),
27.       Variant_New is Variant + 1,
28.       get_variant(Variant_New, MaxVariant, PropertyAccess, Method, ParameterList,
29.                 ObjectInList, ObjectOutList).
```

Code 4 - Getting Variants with op/5 and test/4

13.2 Step 1: Getting Information

In the first step of the Composition Process, the information of the S-Model has to be taken into the C-Model. Information of the Service Request (SRe), which started the Composition Process, is used for the creation of Actor Service Offers (ASO). The second basic data structure of the C-Model, the Service Offer (SO), is built with the information of the Service Template (ST) and a current Service Offer Export Record (SOER).

The building of these data structures includes the construction of Property Objects and their association with the Service Properties, as discussed in the previous section. The Char Property Class ($CharProperty^{Class}$) (cf. Definition 4) given in the declaration of the Service Property as

Char Property in the semantic context of a Semantic Characteristic is used for the construction of the Property Object. The rule (Prop) shows that for all Char Properties (Char_Property) declared in the referenced Semantic Characteristics a describing entry $\left(\text{Char}^{\text{Ref}}, \text{Name}, \text{Prop}^{\text{ORef}}\right)$ and a Property Object is created. This new Property Object is saved in a dictionary. The entry of this dictionary of Property Objects is referenced through so-called object reference of the Service Property ($\text{Prop}^{\text{ORef}}$). The resulting set of describing entries is called a set of Property Descriptions or the information about Service Properties in the C-Model.

$$(\text{Prop})\frac{\text{Char}^{\text{Ref}} \in \left(\text{Char}^{\text{ref}}\right)^{\text{set}}, \text{Char\_Property} \in \text{Char\_Property}^{\text{ Set}}}{(\text{Char}^{\text{Ref}}, \text{Name}, \text{Prop}^{\text{ORef}}), \text{Property\_Object}} \quad \text{if}$$

$$\text{Char\_Property}^{\text{ Set}} \text{ is in Char referenced through Char}^{\text{Ref}},$$
$$\text{Char\_Property} = (\text{Name}, \mathit{CharProperty}^{Class})$$
$$\text{Property\_Object points to implementation instance of } \mathit{CharProperty}^{Class}$$
$$\text{Prop}^{\text{ORef}} \text{ is a reference to a Property Object in a dictionary of Property Objects}$$

A few general remarks to the keeping of Property Objects inside of the data structures of the C-Model are worth to be made. The lists of Service Properties hold only references to the entries of a dictionary of Property Objects. In the definitions of the data structure, these references to the object dictionary are signed with “ObjectRef” or “ORef”. This allows a method handling as introduced in the last section. The main purpose of this method handling was the achieving of backtracking support despite the use of objects. The main idea was the creation of Property Objects in order to get a handle to a new implementation instance of a Property Class, which would incorporate the new (internal) constraints. With the introduction of references to an object dictionary the replacement of the Property Objects can be done without losing the associations with their Service Properties. In future research, these references could be used, in order to introduce a kind of unification on the level of Service Properties.

The object dictionary is indexed with the Property References as introduced by Definition 15. It is kept inside of the data structures (called $\text{Object}^{\text{Set}}$), in order to avoid the use of assert/1 and retract/1 predicates in the declarative environment (cf. Prolog manual [Car2009]), since these predicates are not really declarative. In this way, the Property Objects and their implementation instances can be cleaned up orderly after the life time of the data structure. In the following sub-sections the building of the Service Offer and the Actor Service Offer are described more closely. The section concludes with the description of Value Constraints in the C-Model.

13.2.1 Service Offer

The Service Offer has the same structure like the Service Template: a Common Part, several Service Modes, and several Service Ports per Service Mode (cf. Definition 22 and Definition 8). It is built from the Service Template and a current SOER, which can be always received as a fitting pair from the responsible as well as publishing FA. The reference to the FA is in both data structures (cf. Definition 8 and Definition 9). A FA can declare with a new SOER that all data based on an older one is invalid. Therefore, the Common Part of a SO contains a reference to the SOER, in order to retrace the validity of its data.

Definition 22. Service Offer (SO)

$\mathrm{SO} \equiv (\mathrm{SO}^{\mathrm{ID}},\ \mathrm{SOER}^{\mathrm{Ref}},\ \ \mathrm{SM}^{\mathrm{Set+}}, (\mathrm{GCh}^{\mathrm{Ref}})^{\mathrm{Set}}, \mathrm{SCommonProperty}^{Set}, \mathrm{SO\text{-}Env},\ \mathrm{Object}^{\mathrm{Set}})$
$\mathrm{SM} \equiv (\mathrm{SM}^{\mathrm{ID}}, \mathrm{SP}^{\mathrm{Set+}}, (\mathrm{GCh}^{\mathrm{Ref}})^{\mathrm{Set}}, \mathrm{SModeProperty}^{Set}, \mathrm{SO\text{-}Env})$
$\mathrm{SP}\ \equiv (\mathrm{SP}^{\mathrm{ID}}, (\mathrm{CCh}^{\mathrm{Ref}})^{\mathrm{Set}},\ \mathrm{SPortProperty}^{Set}, \mathrm{SO\text{-}Env})$

$\mathrm{SCommonProperty} \equiv\ (\mathrm{GCh}^{\mathrm{Ref}}, \mathrm{PropName}, \mathrm{Property}^{\mathrm{ObjectRef}})$
$\mathrm{SModeProperty}\ \ \ \equiv\ \left(\mathrm{GCh}^{\mathrm{Ref}}, \mathrm{PropName}, \mathrm{Property}^{\mathrm{ObjectRef}}\right)$
$\mathrm{SPortProperty}\ \ \ \ \equiv \left(\mathrm{CCh}^{\mathrm{Ref}}, \mathrm{PropName}, \mathrm{Property}^{\mathrm{ObjectRef}}\right)$

$\mathrm{SO\text{-}Env} \equiv (\mathrm{Va\text{-}Co}^{\mathrm{Set}}, \mathrm{Ex\text{-}Co}^{\mathrm{Set}}, \mathrm{SO\text{-}Option\text{-}Slot}^{\mathrm{Set}})$

Service Offer (SO) consists of:

- Name of Service Offer ($\mathrm{SO}^{\mathrm{ID}}$)
- Reference to the ST ($\mathrm{ST}^{\mathrm{Ref}}$) in referenced SOER
- Reference of the exporting FA ($\mathrm{FA}^{\mathrm{Ref}}$) in referenced SOER
- Set of Service Modes ($\mathrm{SM}^{\mathrm{Set+}}$)
- Set of Service Ports ($\mathrm{SP}^{\mathrm{Set+}}$)
- Set of References of General Characteristics ($(\mathrm{GCh}^{Ref})^{\mathrm{Set}}$)
- Set of References of Compatibility Characteristics ($(\mathrm{CCh}^{Ref})^{\mathrm{Set}}$)
- Environments (SO-Env) with sets of Value Constraints (Va-Co), Exchange Constraint (Ex-Co), and Option-Slots (SO-Option-Slot)
- Set of objects ($\mathrm{Object}^{\mathrm{Set}}$) for access of instances of the Property Classes
- Sets of Service Properties distinguished by the part of the data structure (common, mode, and port) The Service Properties are defined through their Semantic Characteristic, name of property as defined in the characteristic, and the reference of the object holding the handle to the current instance of the Property Class

The reference of the object (ObjectRef) is related to the set of objects, which is currently indexed through the information introduced as the Property Reference in Definition 15.

Only the Service Modes and Service Ports of the ST, which were declared as valid Service Mode and Service Port, respectively (Valid-SM and Valid-SP in Definition 9), are part of the SO data structure. The building of the environments depends on Option-Slots introduced in the tables Table 13 and Table 14 for the different parts of the data structure. For example, depending on the "overwrite" Option-Slot, not the complete information in the environment description of the ST is taken over into the fitting environment descriptions of the SO.

The constraints of the S-Model (cf. section 10.4) have to be transferred into the C-Model, in order to deal with Property Objects. Since in the general algorithms of the Composition Process, the constraints are applied in the context of certain steps, these transfers and the application of the constraints are described in the context of their steps. Variants of the Composition Process are discussed in the evaluation chapter.

In closer comparison to the data structure of a ST, the data structure of the SO contains additionally the (closer) description of the Service Properties. Every part has its own set of description entries about its Service Properties created through rule (Prop): $\text{SCommonProperty}^{\text{Set}}$, $\text{SModeProperty}^{\text{Set}}$, and $\text{SPortProperty}^{\text{Set}}$.

The information about the Service Property consists of (1) a reference to the Semantic Characteristic, in which context the Service Property was declared as Char Property, (2) the name of the Service Property, and (3) a reference to its Property Object. The Common Part contains the referenced list/dictionary of the Property Objects used in the SO ($\text{Object}^{\text{Set}}$). The Semantic Characteristics appearing in the sets of property information are additionally listed in a way as used in the ST description. As usual the Service Ports of the SO contains the references to the Compatibility Characteristics, which are used for the compatibility description of the Service Offer to another Service Offer or a Service Request (cf. section 10.3). The references point to the ontological repository of the Semantic Characteristic as introduced in section 9.1.

13.2.2 Actor Service Offer (ASO)

The Composition Model transfers a Service Request into one or more so-called Actor Service Offers (ASO). The duality of an ASO to a SO was discussed in the introduction to the C-Model (chapter 12). Like the SO has the ASO a dictionary for the Property Objects and sets of Request Properties holding each a reference to its current Property Object. The parts of the ASO have also separate sets of descriptions of Request Properties (RCommonProperty, RModeProperty, and RPortProperty) created through rule (Prop). The Property sets of the Request Ports like the sets of Service Properties in the Service Ports of an SO are useful for the building of sets of Comparable Properties during the step of discovery of principally compatible Service Offers (cf. Definition 12 and Definition 25). These sets of Comparable Properties are later used for the building of Merge Constraints.

Each Client Request leads to the creation of a data structure, which is similar to the data structure of a Service Mode of a Service Offer. As a Service Client is seen as an **actor**, the generated data structure is called a **Request Mode** (**RM**) of an **Actor Service Offer** (**ASO**) (cf. Definition 23). Through the references to the Service Request (SRe^{Ref}) and the Client Request ($\text{Client-Request}^{\text{Ref}}$), the Request Agent (ReA) and the Personal Agent (PA) can be accessed. The

Definition 23. Actor Service Offer (ASO)

$\text{ASO} \equiv (\text{ASO}^{\text{ID}},\ \text{SRe}^{\text{Ref}},\ \text{RM}^{\text{Set+}}, (\text{GCh}^{\text{Ref}})^{\text{Set}}, \text{RCommonProperty}^{\text{Set}}, \text{ASO-Env}, \text{Object}^{\text{Set}})$
$\text{RM} \equiv (\text{RM}^{\text{ID}}, \text{Client-Request}^{\text{Ref}}, \text{RP}^{\text{Set+}}, (\text{GCh}^{\text{Ref}})^{\text{Set}}, \text{RModeProperty}^{\text{Set}}, \text{ASO-Env})$
$\text{RP} \equiv (\text{RP}^{\text{ID}}, (\text{CCh}^{\text{Ref}})^{\text{Set}},\ \text{RPortProperty}^{\text{Set}}, \text{ASO-Env})$

$\text{RCommonProperty} \equiv (\text{GCh}^{\text{Ref}}, \text{Name}, \text{Property}^{\text{ObjectRef}})$
$\text{RModeProperty} \equiv (\text{GCh}^{\text{Ref}}, \text{Name}, \text{Property}^{\text{ObjectRef}})$
$\text{RPortProperty} \equiv (\text{RCh}^{\text{Ref}}, \text{Name}, \text{Property}^{\text{ObjectRef}})$

$\text{ASO-Env} \equiv (\text{Va-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}}, \text{ASO-Option-Slot}^{\text{Set}})$

Actor Service Offer (ASO) consists of:

- Name of Actor Service Offer (ASO^{ID})
- Reference to the Service Request (SRe^{Ref})
- Reference of the requesting Agent (ReA^{Ref}) in referenced SRe
- Reference of Personal Agent (PA^{Ref}) of Service Client in referenced Client-Request
- Set of Request Modes ($\text{RM}^{\text{Set+}}$)
- Set of Request Ports ($\text{RP}^{\text{Set+}}$)
- Set of References of General Characteristics ($(\text{GCh}^{Ref})^{\text{Set}}$)
- Set of References of Request Characteristics ($(\text{RCh}^{Ref})^{\text{Set}}$)
- Environments (SO-Env) with sets of Value Constraints (Va-Co), Exchange Constraint (Ex-Co), and Option-Slots (ASO-Option-Slot)
- Set of objects ($\text{Object}^{\text{Set}}$) for access of instances of the Property Classes
- Sets of Request Properties distinguished by the part of the data structure (common, mode, and port) The Request Properties are like Service Properties defined through their Semantic Characteristic, name of property as defined in the characteristic, and the reference of the object holding the handle to the current instance of the Property Class

The reference of the object (ObjectRef) is related to the set of objects, which is currently indexed through the information introduced as the Property Reference in Definition 15.

Request Ports of the Client Request become Request Ports of its RM in the ASO. The transfer of the data is strait forward, due to the dual names of the elements.

The Common Part of the Service Request, which is used for General Characteristics and constraints referring to several Client Requests or other parts of the resulting Composite Service, becomes a self-contained part of the Composite Service Structure (CompSt) (cf. Definition 24) in the following initialisation step of the Composition Process. The Request Properties and their General Characteristics in the common part of the ASO ($(\text{GCh}^{\text{Ref}})^{\text{Set}}, \text{RCommonProperty}^{\text{Set}}$) have currently no direct initialisation. In the future, they will take over data of the Actor Service Template (AST), which is already referenced in the Client Requests of the Service Request (SRe, cf. Definition 19). The AST will be used for the learning of user feedback for a specific actor, which can be represented through a Personal Agent (PA) in the application environment. The learnt information could be stored in the Request Properties of General Characteristics, which are visible to every Request Mode by keeping them in the Common Part of the ASO.

The Compatibility Characteristics of a Service Request are all Request Characteristics (cf. section 9.1). Thus, each Client Request and its Request Mode of the ASO become associated with a certain user group. Eventually, an intersection of the user groups associated with a set of Request Characteristics has to be built. (A tool with plausibility checks for the designing of Service Descriptions/Service Requests with Semantic Characteristic and the constraints arising from their semantic descriptions is future research (cf. chapter 22).)

In Example 16, **Alternative Client Requests** were discussed. It can be debated, how to represent the several request of the Service Client in the Service Request or the resulting ASOs. Normally, the resulting data-structure of a Client Request is one ASO with one Request Mode having possibly several Request Ports, each one standing for one Service Request of the potential Service Client. Therefore, the three whished bookings in the example could be described as three separately requested services through three Request Ports.

However, ACTAS offers the option to create several Request Modes of an ASO, when Alternative Client Requests exist. This might be the case, when the client is not sure about the breakdown of the services. In section 11.1 - Service Request (SRe), a way of posing Alternative Client Requests was proposed. Through an ASO with several Service Modes, the alternatives could be tested. Here is again an obvious duality to a SO: as the principal compatibility selects one compatible Service Modes from the alternative Service Modes, the Composition Process has to select one Request Mode of the ASO, in order to initialise its data structures and to begin.

Coming back to the given example and looking at the other extreme: one Client Request could demand that the three bookings are delivered as one service like a packet (possibly through a booking service). In this case, the Client Request is transferred into a Request Mode with only one Request Port holding all three Request Characteristics. It is future research to allow several Request Modes and a dynamic adapting of Request/Service Modes of (Actor) Service Offers, in order to achieve a higher re-active behaviour of ACTAS.

13.3 Step2: Initialisation of the Composite Structure

The primary goal of the algorithm of the Composition Agent (CoA), i.e. of the Composition Process at all, is the detection of a Composite Service. In other words: The building of a Composite (Service) Structure (CompSt) (cf. Definition 24), which does not contain any Open Ports or unsolved constraints (Value, Merge, and Exchange Constraints). During the Composition Process, the CompSt is extended to CompSt-plus (cf. Definition 25), in order to keep temporary information necessary for the Composition Process. An Open Port is a Request Port or Service Port of the CompSt-plus, which is not yet composed with another one. The composition algorithm of the CoA can deal with the Open Ports in different ways according to the policies of the application environment of its ReA. The advantages and disadvantages of these alternatives of composition algorithms are discussed in the evaluation. The Composite Structure and its extension are introduced in the first sub-section.

The CompSt-plus and its embedded CompSt have to be initialised, in order to start the Composition Process. This initialisation is done with the ASOs built from the Client Requests of the SRe in the last step. The Common Part of the Service Request is also used for the initialisation. The Request Ports of the ASOs become the first Open Ports.

In the end of the preceding section, Alternative Client Requests and their resulting in possibly several Request Modes of an ASO were discussed. Even when the current version of ACTAS does not propagate the support of several Request Modes of an ASO, this leads to a common challenge of the Composition Process – the selection of a Service Mode/Request Mode. For ASOs with several Request Modes, the CoA would have to select one Request Mode of each built ASO, in order to test just the fulfilment of these specific Client Requests. The principal compatibility, outlined as step 3 of the Composition Process in the next section, leads to a selection of a Service Mode. Therefore, one sub-section of the current section is concerned with the taking over of information of an ASO or SO with a selected Request Mode or Service Mode into the Composite Structure. This includes the consideration of Value Constraints, since it does not make any sense to go ahead with the Composition Process, when the Value Constraints of a selected mode cannot be satisfied.

13.3.1 The Composite (Service) Structure (CompSt)

The Composite (Service) Structure gets a more complicated than the earlier data structures, since it has to hold the resulting data of a Composite Service, i.e. it does not contain Service/Request Ports as interfaces any longer, but so-called merged Service Ports (Merged Ports for short, Merged-SP in Definition 24 - Composite Service Structure (CompSt)). Merged Ports are Service/Request Ports, which were composed with the rules of principal compatibility (cf. Definition 11), and which hold Comparable Service Properties (cf. Definition 12) tested with their Merge Constraints (cf. Definition 16). A Merged Port holds references to the "merged" Service/Request Ports and a set of the "merged" Comparable Service Properties ($\text{MeProperty}^{\text{Set}}$ in Definition 24). Each MeProperty entry contains descriptions of the two Comparable Properties (cf. Definition 12), i.e. their Compatibility Characteristic and name as well as references to their Property Objects. The entry also contains an association of the Comparable Properties with their Merge Property Object created through the Merge Constraint (cf. section 13.5 - Step 4: Checking of Merge Constraints). The Merge Property Classes of these Merge Property Objects were linked with the Char Properties in the Compatibility Characteristics at the time point of declaration (cf. Definition 4).

The references to the Service/Request Ports ($\text{Cl-SP}^{Ref}, \text{Se-SP}^{\text{Ref}}$) in Merged-SP distinguish between client and server side for directed compositions (BOOLEAN "Directed" is set for a directed composition.). The same order can be found for the references of the Property Objects ($\text{Cl-Prop}^{ORef}, \text{Se-Prop}^{ORef}$) in the above introduced entries for the description of the Merge Properties (MeProperty) of the merged Service Ports (Merged-SP). For non-directed compositions, the distinction between server and client side is irrelevant, but the fixed order helps again, in order to deal with the view information of the extended reference of a Service Property (cf. Definition 17) in a non-ambiguous way. The extended access of a Service Property with the view information becomes necessary for the solving of Exchange Constraints, which can describe in their "ExchangeProperties" data (cf. Definition 18) this elaborated retrieving of property data. In the C-Model, the Exchange Constraints use the methods "export" and "import", in order to get hold of the right Service Properties and to have discrete Property Objects for the handling of the elements of the Exchange Constraints called by their "ExNames" (cf. Definition 18, rules (Export) and (Import), and section 13.6 - Step 5: Checking of Exchange Constraints).

Definition 24. Composite Service Structure (CompSt)

$\text{CompSt} \equiv (\text{CompSt}^{\text{ID}}, \text{Selected-SM}^{\text{Set+}}, \text{ComProperty}^{\text{Set}}, \text{Merged-SP}^{\text{Set}}, \text{Object}^{\text{Set}})$
$\text{Selected-SM} \equiv (\text{Sel-ASO-RM} \mid \text{Sel-SO-SM})$
$\text{Merged-SP} \equiv (\text{Mer-SP}^{\text{ID}}, \text{Cl-SP}^{Ref}, \text{Se-SP}^{\text{Ref}}, \text{Directed}, \text{MeProperty}^{\text{Set}})$

$\text{Sel-SO-SM} \equiv (\text{SM}^{\text{ID}}, \text{SOER}^{\text{Ref}}, \text{SO-SM}^{\text{Ref}}, \text{Res-Info}, \text{ModeProperty}^{\text{Set}}, (\text{Merged-SP}^{\text{Ref}})^{\text{Set}})$
$\text{Sel-ASO-RM} \equiv (\text{RM}^{\text{ID}}, \text{SRe}^{\text{Ref}}, \text{ASO-RM}^{\text{Ref}}, \text{ModeProperty}^{\text{Set}}, (\text{Merged-SP}^{\text{Ref}})^{\text{Set}})$

$\text{ComProperty} \equiv (\text{GCh}^{\text{Ref}}, \text{PropName}, \text{Prop}^{ORef})$
$\text{ModeProperty} \equiv (\text{GCh}^{\text{Ref}}, \text{PropName}, \text{Prop}^{ORef})$
$\text{MeProperty} \equiv (\text{CCh}^{\text{Ref}}, \text{PropName}, \text{Me-Prop}^{\text{ORef}}, \text{Cl-Prop}^{ORef}, \text{Se-Prop}^{ORef})$

Composite Structure (CompSt) consists of:

- Common Information about the requested Composite Service (Common)
- Set of selected Service Modes (SM) (at least one element) ($\text{Selected-SM}^{\text{Set+}}$)
    - selected Service Mode of SO (Sel-SO-SM)
    - selected Service Mode of ASO (Sel-ASO-SM)
- Set of merged Service Ports as results of Merge-Constraints ($\text{Merged-SP}^{\text{Set}}$)
- Set of Objects ($\text{Object}^{\text{Set}}$)
- MeProperty contains descriptions of Comparable Properties (cf. Definition 12)) and of the Merge Property of the Merge Constraint applied on the two Comparable Properties

Like the data structures of SO and ASO, the CompSt holds a dictionary of Property Objects ($\text{Object}^{\text{Set}}$). Service Properties in the semantic context of General Characteristics (GCh) can be found in the Common Part of CompSt ($\text{ComProperty}^{\text{Set}}$) and in the selected Service Modes (Selected-SM) ($\text{ModeProperty}^{\text{Set}}$). They are based on Char Property Classes and the rule (Prop) was applied, in order to create their Property Objects and the entries of their Property Description sets. The Merge Service Properties ($\text{MeProperty}^{\text{Set}}$) of the merged Service Ports (Merged-SP) also reference Merge Property Objects created during the application of the Merge Constraints (cf. section 13.5 - Step 4: Checking of Merge Constraints). In the application of the Exchange Constraints, methods of Exchange Property Objects are used.

The selected Service Mode can either come from a Service Offer (SO) or an Actor Service Offer (ASO). Therefore, the data structure of CompSt allows two alternatives (Sel-SO-SM, Sel-ASO-RM). The selected Service Mode of a SO contains references to its SOER and to the SM of the SO. Through this information, the Facility Agent (FA) can be retrieved. ACTAS supports a validity checking of the selected Service Mode, i.e. the FA could be asked whether the SOER is still valid. Additionally, the mentioned data can support the resource management and the reservation of the Component Service. This is part of the Post-Processing and is discussed in section 13.7. The selected Request Mode of an ASO (referenced through ASO-$\mathrm{RM}^{\mathrm{Ref}}$) holds information about the Service Request (SRe). Thus the Client-Request and the involved agents (ReA and PA) are accessible from CompSt. These links can be used in the Post-Processing negotiations.

13.3.2 Working with the Extended Composite (Service) Structure (CompSt-plus)

After the building of Actor Service Offers (ASO) from the Service Request, the Composition Process can commence. For the processing of the composition the Composition Structure CompSt has to be extended. The extended data structure is called **CompSt-plus** (cf. Definition 25), which contains the CompSt and additional environment information as well as elements necessary for keeping of process information. In the following, these fields are described more closely, in order to reach an understanding of the working with the CompSt-plus data structure.

Definition 25. Composite Service Structure (CompSt-plus)

$$\begin{aligned} \text{CompSt-plus} &\equiv (\text{CompSt}, \text{Va-Co}^{\text{Set}}, \text{Me-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}}, \text{Open-SP}^{\text{Set}}) \\ \text{Open-SP} &\equiv (\text{Open-SP}^{\text{ID}}, \text{SP}^{\text{Ref}}, \text{Option-Slot}^{\text{Set}}, \text{depth}, \left(\text{CCh}^{\text{Ref}}\right)^{\text{Set}}, \text{OpProperty}^{\text{Set}}) \\ \text{OpProperty} &\equiv \left(\text{CCh}^{\text{Ref}}, \text{PropName}, \text{Property}^{\text{ObjectRef}}\right) \end{aligned}$$

Extended Composite Structure (CompSt-plus) consists of:

- Composite Structure (CompSt) (cf. Definition 24)
- Set of Value-Constraints ($\text{Va-Co}^{\text{Set}}$)
- Set of Merge-Constraints ($\text{Me-Co}^{\text{Set}}$)
- Set of Exchange-Constraints ($\text{Ex-Co}^{\text{Set}}$)
- Set of open Service-Ports with Option-Slot for principal compatibility
- Directed (a BOOLEAN value) says if a client-server relationship is given.
- Set of Service Properties in the Compatibility Characteristics of the Open Port

First of all, it has to be clarified, what information the Composite (Service) Structure (CompSt) data structure, introduced in Definition 24 and the previous sub-section, can keep and how the information of the Service Request is stored. The Fig. 36 - Visibility of Properties in CompSt – shows an arrangement of the Semantic Characteristics appearing in the different parts of a Composite Service. A Compatibility Characteristic in the merged Service Ports can define constraints only for its own Service Properties and the “merged” Service Properties. The General

Characteristics in the Service Modes can introduce as "building blocks" Exchange Constraints (cf. Example 15), which can also concern the Service Properties of their Service Ports. Finally, the General Characteristics of the Common Part of the Composite Structure, which have their origin in the Common Part of the Service Request, can introduce Exchange Constraints for Service Properties of the whole Composite Structure. This extension of the scope of Service Properties, which can be addressed in the Exchange Properties term of Exchange Constraints (cf. Definition 18), is called the "visibility" of Service Properties.

The data of the Service Properties is wrapped in their implementation instances and accessible through their Property Objects. Accordingly to the figure, the information of CompSt can be structured as follows:

- Service Properties with information about the whole Composite Service ($\mathsf{ComProperty}^{\mathsf{Set}}$) initialised through the Common Part of the Service Request
- The selected Service Modes of (Actor) Service Offers ($\mathsf{Selected\text{-}SM}^{\mathsf{Set}}$).
- Service Properties with information about the Service Modes ($\mathsf{ModeProperty}^{\mathsf{Set}}$), satisfying Value and Exchange Constraints
- The Component Services with their merged Service Ports (Merged-$\mathrm{SP}^{\mathrm{Set}}$)
- Service Properties with information about the Component Services, fulfilling Value, Merge, and Exchange constraints (Merge Constraints), i.e. the information of the Service Properties results from the matching of request and offering of the Component Services.

The CompSt-plus data structure holds beside the CompSt itself a set of **Open Port**s ($\mathsf{Open\text{-}SP}^{\mathsf{Set}}$ in Definition 25). These are Service/Request Ports, which are not yet composed with (principally) compatible/matching ones. The organisation and the use of the set of Open Ports is a main criterion of the algorithm of the Composition Process done by the CoA (cf. section 13.4 - Step 3: Service Discovery and Principal Compatibility and evaluation chapter). An Open Port takes over the set of references to Compatibility Characteristics and the Option-Slots held in the Service/Request Port (cf. SO (Definition 22) and ASO (Definition 23)). This information is relevant for the Service Discovery of principally compatible Service Offers (cf. Definition 11) and used for the building of the Trading Request (TRe).

The environment description (Env) in the CompSt-plus data structure is not any longer distributed over the different parts (Common Part, Mode, and Ports) as in previous data structures, but concentrated in the Common Part. This means that the constraints have to be collected for the selected Service Modes, and translated into the declarative environment of the C-Model (i.e. they have to use the Property Objects). The constraints are applied during the Composition Process. The general Composition Process, introduced in this chapter, applies the Value Constraints directly on the Service Properties of the selected Service Modes, in order to avoid inconsistent Service Descriptions as early as possible in the Composition Process. The process of selecting a Service Mode and the application of the Value Constraints is covered in the next sub-sections.

During the Composition Process, the merged Service Port element (Merged-SP) is keeping the Service/Request Ports recognized as principally compatible. The set of merged Service Properties ($\mathtt{MeProperty}^{\mathrm{Set}}$) collects the descriptions of Comparable Properties as discussed in the previous sub-section. This information and the BOOLEAN field "Directed" are used for the building of Merge Constraints (cf. section 13.5 - Step 4: Checking of Merge Constraints). For a directed composition, the merged Service Port element has specific fields for the Service Port of the client Side (also called **Client Port** or **OUT Port**) as well as for the Service Port of the server side (also called **Server Port** or **IN Port**). This distinction between client and server side is continued for the property description entries called MeProperty.

13.3.3 Initialisation of CompSt-plus and Selection of a Service Mode

The following steps of initialisation of the extended Composite (Service) Structure (CompSt, Definition 24, CompSt-plus Definition 25) have to be performed and are discussed in this sub-section:

- The set of Property Descriptions in the Common Part of CompSt ($\mathtt{ComProperty}^{\mathrm{Set}}$) have to be taken from the General Characteristics in the Common Part of the Service Request (SRe-Common in Definition 19).
- One Request Mode of each ASO built from the Client Requests of the Service Request has to be selected, in order to become the first selected Service Mode (Selected-SM, Sel-ASO-RM) of CompSt.

Following the "building blocks" concept discussed in the S-Model, the administrator of the application environment, responsible for the design of Service Requests, can use the General Characteristics and environment descriptions in the Common Part of the Service Request (SRe-Common in Definition 19) for the definition of constraints and general information valid for several Client Requests or greater parts of the resulting Composite Structure. In the figures of Service Compositions (e.g. Fig. 50 and Fig. 38), this Common Part of the Composite Structure and its General Characteristic(s) is shown through a discrete yellow rectangle.

Fig. 36 - Visibility of Properties in CompSt - shows in the top row the General Characteristics of the Common Part of CompSt, which had their origin in the Common Part of the Service Request (SRe-Common in Definition 19). In fact, CompSt contains the Service Property Descriptions of these General Characteristics in its Common Part ($\mathtt{ComProperty}^{\mathrm{Set}}$ in Definition 24). They are built as a first initialisation step of the CompSt-plus data structure through the rule (Prop). The rule takes the set of General Characteristics in SRe-Common for the building of the Service Property Description $\mathtt{ComProperty}^{\mathrm{Set}}$ and creates the Property Objects referenced in the entries of the Service Property Description.

The Exchange Names (ExNames) of the Exchange Constraints (cf. Definition 18), appearing in the Common Part, can be associated with Service Properties of any part of CompSt, i.e. the Service Properties could belong to the Common Part, in the selected Service Modes, or the Service Ports due to the visibility of the Service Properties illustrated in Fig. 36. The associations between ExNames and Property-References are done through the Exchange Properties term (cf. Definition 18). However, the provision of dynamic reference formalism, which allows the use of the visibility of Service Properties inside CompSt properly, is still topic of research, since selected Service Modes of the Composite Service are unknown at the time point of the Service Request. The reference of properties described through Definition 15 allows a direct reference of the Request Properties given in the Service Request, i.e. references to the Service Properties of the ASOs built with the SRe are possible. This is sketched in Case Study 2: Distribute Feature Composition (DFC): the GCh "Feature" is used for the checking of Distributed Feature Composition through links with the Service Properties of Compatibility Characteristics (also called "Feature") holding the feature descriptions wished by the two involved Service Clients.

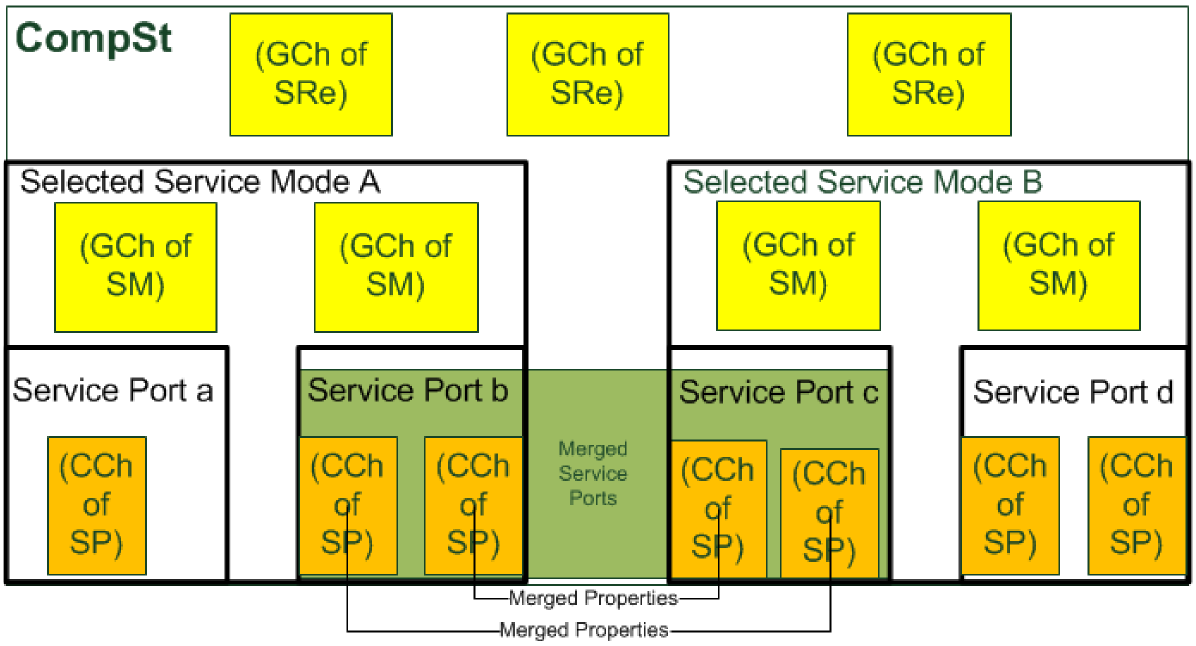


Fig. 36 - Visibility of Properties in CompSt

In Fig. 38 - Technical Service shown with principal compatibility, the selected Service Modes of the gateway service and of the IP-Telephony service hold a General Characteristic called "Reliability". In one of their Service Ports, a Compatibility Characteristic called "H.323 Reliability" is "visible". This Compatibility Characteristic might have ensured the existence of reliability checking on the base of the standard H.323. The General Characteristic "Reliability" in the selected Service Mode could realise an extended testing of service reliability through the definition of Exchange Constraints accessing Service Properties of the visible Compatibility Characteristic. Thus, there should be a "works with" relationship between the mentioned GCh and CCh in the ontological repository of the Semantic Characteristics (cf. section 9.1).

The described scenario can be extended through a General Characteristic “Reliability” in the Common Part of the CompSt, given through the Common Part of the Service Request at the time point of the Service Request. This General Characteristic could test reliability through Exchange Constraints for the whole Composite Service. For this purpose, its Exchange Properties term could interlink Service Properties of all “Reliability” General Characteristics kept in the selected Service Modes of CompSt. A proposed dynamic reference of the Service Properties could just contain a general reference to the General Characteristic “Reliability”, which would be interpreted at the time point of application of the Exchange Constraint. At this time point, the Composite Structure should be complete and all selected Service Modes are visible. Thus, all Service Properties of General Characteristics “Reliability” appearing in the selected Service Modes could be accessed. This means that the number of involved ExName elements of the Exchange Constraints would dynamically grow with the number of General Characteristics of this kind appearing in the selected Service Modes of CompSt.

$$\text{(InitialiseStep2)}\ \frac{\text{As long as there are ASOs left with no selected RM}}{\begin{array}{c}\text{select RM of next ASO}\\ \text{do for selected RM rule (Sel-ASO-RM)}\\ \forall \text{RP} \in \text{RP}^{\text{Set}} \text{ of RM do rule (Add-Open-SP)}\end{array}}$$

In section 13.2 - Step 1: Getting Information, it was described, in which way each Client-Request was transformed into one ASO. The second action of the initialisation of the CompSt-plus data structure is the transfer of the data of the ASO as described through rule (InitialiseStep2). In the current version of ACTAS, the number of Request Modes of an ASO was limited to one. In a future version, the CoA will select a Request Mode through a backtracking algorithm, since every Request Mode will stand for an Alternative Client Request (cf. Example 16). The data-structure of an ASO already contains a set of Service Property Descriptions with their references to Property Objects. The Property Objects and the sets of Property Descriptions can directly be taken over (rule (Sel-ASO-RM)). The information of the Request Ports of the selected Request Mode is saved in the set of Open Ports ($\text{Open-SP}^{\text{Set}}$); these initialisation actions are described with rule (Add-Open-SP).

$$\text{(Sel-ASO-RM)}\ \frac{\begin{array}{c}(\text{Va-Co}^{\text{Set}}, \text{Me-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}})_{\text{CompSt-plus}}, \text{Object}^{\text{Set}}_{\text{CompSt}}, \text{ASO-Env}_{\text{ASO}}, \text{ASO-Env}_{\text{RM}},\\ \text{RCommonProperty}^{\text{Set}}, \text{RModeProperty}^{\text{Set}}, \text{Object}^{\text{Set}}_{\text{ASO+RM}}\end{array}}{\begin{array}{c}\left((\text{Va-Co}^{\text{Set}})', \text{Me-Co}^{\text{Set}}, (\text{Ex-Co}^{\text{Set}})'\right)_{\text{CompSt-plus}}, \left(\text{Object}^{\text{Set}}_{\text{CompSt}}\right)'\\ \text{SRe}^{\text{Ref}}, \text{ASO-RM}^{\text{Ref}}, \text{ModeProperty}^{\text{Set}}\end{array}}\ \text{if}$$

$$\begin{array}{c}\text{ASO-RM}^{\text{Ref}} \text{ references selected RM of ASO},\\ \left(Object^{Set}_{\text{CompSt}}\right)' = Object^{Set}_{\text{CompSt}} \cup Object^{Set}_{\text{ASO+RM}},\\ (\text{Va-Co}^{\text{Set}})' = \text{Va-Co}^{\text{Set}} \cup getVaCo(\text{ASO-Env}_{\text{RM}} \cup \text{ASO-Env}_{\text{ASO}}),\\ (\text{Ex-Co}^{\text{Set}})' = \text{Ex-Co}^{\text{Set}} \cup getExCo(\text{ASO-Env}_{\text{RM}} \cup \text{ASO-Env}_{\text{ASO}}),\\ \text{ModeProperty}^{Set} = \text{RModeProperty}^{\text{Set}} \cup \text{RCommonProperty}^{Set} \text{ with key } (\text{GCh}^{\text{Ref}}, \text{Name})\end{array}$$

Rule (Sel-ASO-RM) shows that the Service Property Description of a selected Service Mode ($\text{ModeProperty}^{Set}$) is directly the unification of the sets of Service Property Descriptions of the (chosen) Request Mode and the Common Part of the ASO. However, only entries of the Service Property Description of the Request Mode part ($\text{RModeProperty}^{\text{Set}}$) will be included in the union set when entries in the Service Property Description of the Common Part ($\text{RCommonProperty}^{Set}$) exist that reference the same General Characteristic, since (GCh^{Ref},Name) are used like a key. The dictionary of Property Objects $Object^{Set}_{\text{ASO+RM}}$, coming from the Common Part and the Request Mode of the ASO, contains only the Property Objects, which have object references in the newly built set of Property Description of the selected Service Mode ($\text{ModeProperty}^{Set}$). The Value Constraints and the Exchange Constraints in CompSt-plus are extended with the constraints appearing in the environment descriptions of the Common Part and of the Request Mode. In this process, the CoA might interpret Option-Slots found in the environments of the ASO.

An Open Port (Open-SP) is a Service/Request Port of a selected Service Mode of the Composite Structure (CompSt), which is not yet composed with another Service Port of a Service Offer (SO) or Actor Service Offer (ASO). An Open Port of an ASO gets the initialising value zero in its field "depth" because it starts the Composite Structure. In general, the depth value of an Open Port of a selected Service Mode equals the minimal path length of composition of the selected Service Mode as defined in Definition 27. The rule (Add-Open-SP) adds the information of a Service/Request Port as a new Open Port. The Service Property Description of the Service Port ($\text{PortProperty}^{\text{Set}}$) becomes directly the Service Property Description of the Open Port ($OpProperty^{Set}$). The referenced Property Objects ($Object^{Set}_{\text{Port}}$) are added to the ones already known in CompSt ($Object^{Set}_{\text{CompSt}}$). The Value and Exchange Constraints are taken over. The Option-Slots and the references to the Compatibility Characteristics are used for the determination of the principal compatibility.

$$(\text{Add-Open-SP})\ \frac{\begin{array}{c}(\text{Va-Co}^{\text{Set}}, \text{Me-Co}^{\text{Set}}, \text{Ex-Co}^{\text{Set}})_{\text{CompSt-plus}}, \text{Object}^{\text{Set}}_{\text{CompSt}}, \text{Env},\\ (\text{CCh}^{\text{Ref}})^{\text{Set}}_{\text{Port}}, \text{PortProperty}^{\text{Set}}, \text{Object}^{\text{Set}}_{\text{Port}}\end{array}}{\begin{array}{c}((\text{Va-Co}^{\text{Set}})', \text{Me-Co}^{\text{Set}}, (\text{Ex-Co}^{\text{Set}})')_{\text{CompSt-plus}}, (\text{Object}^{\text{Set}}_{\text{CompSt}})'\\ \text{SP}^{\text{Ref}}, \text{Option-Slot}^{\text{Set}}, \text{depth}, (\text{CCh}^{\text{Ref}})^{\text{Set}}_{\text{Op}}, \text{OpProperty}^{\text{Set}}\end{array}}\ \text{if}$$

$$\begin{array}{c}\text{SP}^{\text{Ref}} \text{ references selected SM/RM of SO/ASO},\\ \text{depth=minimal path length of SM/RM (cf. Definition 27)},\\ (\text{Object}^{\text{Set}}_{\text{CompSt}})' = \text{Object}^{\text{Set}}_{\text{CompSt}} \cup \text{Object}^{\text{Set}}_{\text{Port}},\\ (\text{Va-Co}^{\text{Set}})' = \text{Va-Co}^{\text{Set}} \cup (\text{getVaCo(Env)}),\\ (\text{Ex-Co}^{\text{Set}})' = \text{Ex-Co}^{\text{Set}} \cup (\text{getExCo(Env)}),\\ \text{Option-Slot}^{\text{Set}} = \text{getOpSlot(Env)},\\ \text{OpProperty}^{\text{Set}} = \text{PortProperty}^{\text{Set}},\ (\text{CCh}^{\text{Ref}})^{\text{Set}}_{\text{Op}} = (\text{CCh}^{\text{Ref}})^{\text{Set}}_{\text{Port}}\end{array}$$

In Example 17 - Telecommunication with Gateway - a possible result of the initialisation of the Composite Service Structure (CompSt) is shown (Fig. 37). The assumed Client Requests of the Service Request of the example are listed below. The example was chosen, in order to demonstrate directed and non-directed Service Composition. More complex examples of Business Services and (Sematic) Web Services are covered in the evaluation. It is assumed that the Service Request of Example 17 was done on behalf of two Service Clients, who wanted to have Audio Communication. However, it turned out in the Composition Process that their Communication Facilities are connected to different networks. Thus, the use of a gateway becomes necessary (cf. the continuation of the example in Example 18). In the next sub-section, the application of Value Constraints is described.

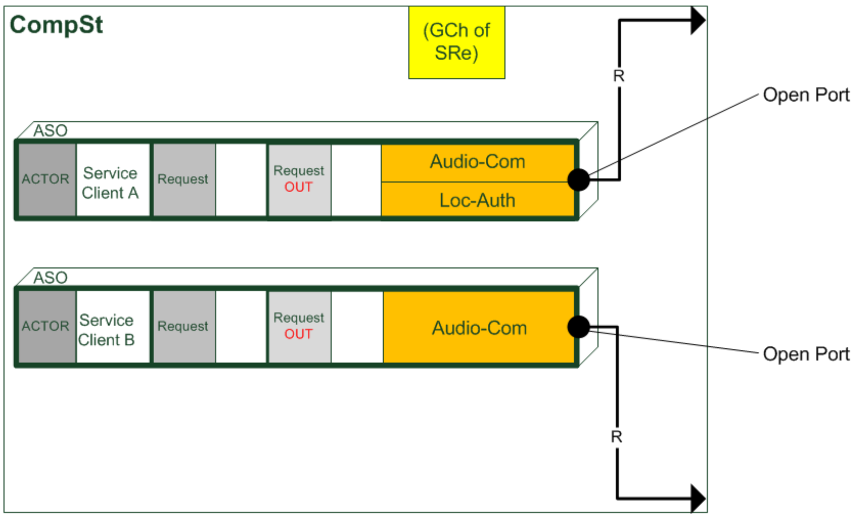


Fig. 37 - Initialised Composite Structure

Example 17 **Telecommunication with Gateway**

This example demonstrates a simple composition of services with directed and non-directed compositions. In order to establish a telephone connection between two users having telecommunication facilities, which are connected to different networks, the Communication Service has to include an IP-Gateway. The two Client Requests have the following sets of Request Characteristics: [Audio-Com, Loc-Auth] and [Audio-Com], respectively, as shown above. The Request Characteristic "Audio-Com" may just look for services offering audio communication like phones. The "Loc-Auth" is assumedly a Request Characteristic, which check the location and authorisation. The likely idea of the combination of these Request Characteristics was to ensure the reachability and accessibility of the communication facility, i.e. a person should for instance use his own mobile when available. In this way, the Personal Agent could also become the Facility Agent. The sketched Service Request will result in two Actor Service Offers shown in Fig. 37. The two ASOs are used for the initialisation of the extended Composition Structure (CompSt-plus). The resulting data is presented with the sets of selected Service Modes and Open Ports (Selected-SM$^{\text{Set}}$ and Open-SP$^{\text{Set}}$).

$$\text{Client-Request}^{Set} \text{ of } \text{SRe}_1 \text{ of Example 17} = \Big[\big(\text{RM}_1, \text{PA}_1^{\text{Ref}}, \text{AST}_1^{\text{Ref}}, \text{RP}_1^{\text{Set}}, [\_], \text{SRe\_Env}_1\big), \big(\text{RM}_2, \text{PA}_2^{\text{Ref}}, \text{AST}_2^{\text{Ref}}, \text{RP}_2^{\text{Set}}, [\_], \text{SRe\_Env}_2\big)\Big]$$

$$\text{RP}_1^{\text{Set}} = \big[(\text{RM}_1, \text{RP}_1), [\text{Audio-Com, Loc-Auth}], \text{SRe\_Env}_3\big]$$

$$\text{RP}_2^{\text{Set}} = \big[(\text{RM}_2, \text{RP}_1), [\text{Audio-Com}], \text{SRe\_Env}_4\big]$$

$$\text{Selected-SM}^{\text{Set}} = \big[((\text{CompSt},\text{RM}_1),\ \text{SRe}_1, (\text{ASO}_1,\text{RM}_1),\ \_,\ \_), ((\text{CompSt},\text{RM}_2),\ \text{SRe}_1, (\text{ASO}_2,\text{RM}_1),\ \_,\ \_)\ \big]$$

$$\text{Open-SP}^{\text{Set}} = \begin{bmatrix} (\text{OP}_1, (\text{ASO}_1,\text{RM}_1,\ \text{RP}_1),\ [reqest, direction(OUT)], 0, [\text{Audio-Com, Loc-Auth}]\ \_), \\ (\text{OP}_2, (\text{ASO}_2,\text{RM}_1,\ \text{RP}_1),\ [reqest, direction(OUT)], 0, [\text{Audio-Com}]\ \_) \end{bmatrix}$$

13.3.4 Value Constraints

Value Constraints restrict the possible values of Service Properties. The Service Designer and the Service Requester used them for the description of the values of the Service Properties of the Service Offer or the Service Request, respectively. They apply the methods of the Char Property Class given with the declaration of the Service Property in the context of a Semantic Characteristic. This Semantic Characteristic is likely for containing Value Constraints for the Service/Char Property, as well, in order to adapt its values to the constraints of its semantic context.

Definition 26. Value Constraint (Va-Co) in C-Model

$$\text{ValueConstraint} \equiv \text{va-co}(\text{Property}^{\text{Ref}}, \text{ValueClause})$$

$$\text{ValueClause} \equiv [clause_1, \cdots, clause_m]$$

$$\text{clause}_k \equiv \text{op}\begin{pmatrix}\text{PropObject}_k, \text{method}_k, [Parameter_1, \cdots, \text{Parameter}_n], \\ [\text{PropObject}_k], [\text{PropObject}_{k+1}]\end{pmatrix}$$

Meaning:

- $n, m, k \in \mathbb{N}$,
- $\text{Property}^{\text{Ref}}$ is a reference to the Service Property (cf. Definition 15) declared with the Char Property Class of the Char Property Objects PropObject_k, PropObject_{k+1}, respectively.
- PropObject_k is the version k of the Service/Char Property Object, which gives access to an implementation instance that granted all constraints originated from the method calls of $clause_1$ to $clause_{k-1}$ (cf. rule (ExOp))
- The ValueClause contains a list of methods calls with op/5 (i.e. semantically a conjunction of the resulting constraints is realised. ($\text{clause}_1 \wedge \cdots \wedge \text{clause}_m$)).
- In a successful case, a PropObject_{m+1} is generated.

Value Constraints are part of the environment descriptions (Env) in a Semantic Characteristic, a Service Description, and a Service Request. The discussion about Option-Slots (cf. section 10.2) showed that Value Constraints in a SOER environment can overwrite the ones in the ST environment, i.e. the Value Constraints of the ST will be simply discarded in the collection process for the building of a Service Offer (SO), when the Option-Slots are set in this way (cf. section 13.2.1).

The CompSt-plus data structure collects the Value Constraints and the other constraints in one set in the Common Part (cf. Definition 25). For the application of Value Constraints, they have to be translated from the informal form of the S-Model (Definition 14) to the form of the C-Model (Definition 26) using Property Objects instead of Property References. That also implies, as Definition 26 shows, the use of the predicate op/5 for the clauses of the Value Constraint. The predicate op/5 was introduced in section 13.1. It calls a method of the Char Property Object associated with the Service Property, in order to create a new Char Property Object, which incorporated successfully the internal constraints coming from the application of the method. The newly Char Property Object is used in the on-going backtracking algorithm. In section 13.1, the rules (ExOp) and (intOp) were introduced, in order to show schematically how the methods of the constructed object are used for the setting of new internal Value Constraints.

$$(\text{Get-Value-Co})\frac{\text{Va-Co}^{\text{Set}}, Property^{Ref}}{\text{va-co}(\text{Property}^{\text{Ref}}, \text{ValueClause})}\ if$$

$$\begin{gathered}
\text{getChar}(\text{Property}^{\text{Ref}}, \text{Char}),\\
\text{Char-Env of Char},\\
\text{ValueClause}_1\text{=translate}\Big(\text{select}\big(\text{PropertyName}, \text{getVaCo}(\text{Char-Env})\big)\Big),\\
\text{ValueClause}_2\text{=translate}\Big(\text{select}\big(\text{Property}^{\text{Ref}}, \text{Va-Co}^{\text{Set}}\big)\Big),\\
\text{ValueClause=ValueClause}_1 \cup \text{ValueClause}_2
\end{gathered}$$

For the application of Value Constraints on the Property Object, the Value Clause (ValueClause) of Definition 26 has to be built. The collecting of Value Constraints from the set of Value Constraints of CompSt-plus, being relevant for a referenced Service Property

```
applyValueConstraint(va-co(PropertyRef,ValueClause)) :-
    translateValueClause(ValueClause, OpSet),
    getCharPropertyObject(PropertyRef,CharPropertyObject),
    applyValueOpSet(OpSet, CharPropertyObject, CharPropertyObjectOut),
    addToObjectSet(PropertyRef, CharPropertyObjectOut).

translateValueClause([Clause|RestClauses], [OP|RestOPs]) :-
    Clause =.. [Method, ParameterList],
    OP = op(A,Method,ParameterList,[A],[CurrentVar]),
    translateValueClause2(RestClauses, RestOPs, CurrentVar).

translateValueClause([],_) :-
    print('error').

translateValueClause2([Clause|RestClauses], [OP|RestOPs], CurrentVar) :-
    Clause =.. [Method, ParameterList],
    OP = op(CurrentVar, Method, ParameterList, [CurentVar], [CurrentVarNew]),
    translateValueClause2(RestClauses, RestOPs, CurrentVarNew).

translateValueClause2([], [], _).

applyValueOpSet([OP|RestOPs], CharPropertyObject, CharPropertyObjectOut) :-
    A = CharPropertyObject,
    OP = op(A, Method, ParameterList, [A], B),
    call(OP),
    applyValueOpSet(RestOPs, B, CharPropertyObjectOut).

applyValueOpSet([], CharPropertyObjectOut, CharPropertyObjectOut).
```

Code 5 - Translation and Application of Value Constraints

($Property^{Ref}$, Definition 14), as well as the building of the Value Clause is shown with rule (Get-Value-Co). First of all, the Value Constraints of the environment description of the declaring Semantic Characteristic are fetched (getVaCo(Char-Env)) (Char-Env, cf. Definition 4). The select/2 function selects the Value Constraints relevant for one specific Service Property. The second set of Value Constraints is directly selected from the set of Value Constraints of the CompSt-plus data structure. In fact, the Value Clause sets are ordered lists of the clauses, which can semantically be interpreted as a conjunction of conditions described through the methods of the Property Class/Object. The translate function creates the use of the Property Objects with op/5. In Code 5 - Translation and Application of Value Constraints, the translation is schematically shown in Prolog with predicate translateValueClause/2. The result of this predicate is a list of op/5 calls; its input is given with the Value Clause as a list of clauses with method calls as introduced in the S-Model. The translation takes advantage of logical variables. Therefore, only the first logical variable has to be unified with the current Property Object, in order to achieve the wished behaviour of Definition 26, the creation of new Property Objects for the backtracking and monotony of the constraints as long as the resulting internal constraints can be successfully incorporated i.e. it still exist a model (cf. section 13.1, rule (intOp)).

13.4 Step 3: Service Discovery and Principal Compatibility

The initialisation of the Composition Process, described in the last section, created a CompSt-plus data structure holding the Request Modes of the ASOs as first selected Service Modes. The Request Ports of these Request Modes were collected as Open Ports. The Service Properties of General Characteristics in the Common Part had their origin in the Common Part of the Service Request. As discussed in the last section, they describe information for the whole CompSt. Every Service Property has an association with a Property Object as a handle to the implementation instance of the Service Property. At last, the constraints of the Service Request were collected in central sets in the Common Part of CompSt-plus. Additionally, the Value Constraints were applied on the Service Properties of the CompSt-plus data structure, in order to exclude inconsistent Service/Request Descriptions as early as possible.

The step 3 of the Composition Process describes its main cycle of actions; it performs the Service Composition and takes advantage of Service Trading for the Service Discovery of new Service Offers as candidates for the Service Composition. The first rule applied in this step is the rule (Sel-OpenSP), in order to determine the next currently selected Open Port (COP for short). The next principal compatible Service Port is searched for this COP. As rule (Sel-OpenSP) states, the COP is deleted as an element of the set of Open Ports after its selection. The algorithm of the Composition Process depends on the smart selection of the next COP. Therefore, this matter is covered in the next two paragraphs and in the evaluation.

The data of the Open Port contains a field called "depth"; its content is defined in Definition 27. This definition keeps in mind that the Composition Process starts with a Service

Definition 27. Graph, Path Length, and depth of Service Composition

$$G_{\text{Composition}} \equiv (\text{SelectedServiceModes, MergedServicePorts})$$

$$\text{StartVertices}(G_{\text{Composition}}) \equiv \text{set of SelectedServiceModes of ASOs built from Client Requests of the Service Request}$$

$$\text{path}_{\text{Composition}}(SM_1, SM_2) \equiv MP_1, MP_2, \ldots, MP_n$$

sequence of MergedServicePorts with
(1) MP_i and MP_{i+1} have a common selected Service Mode
(2) the sequence does not contain a loop
(3) MP_1 has SM_1 as vertex, MP_n has SM_2 as vertex
$(i = 1,\ldots,n-1)$, $n \in \mathbb{N}$

$$\text{length}\left(\text{path}_{\text{Composition}}(SM_1, SM_2)\right) \equiv \text{number of MergedServicePorts of path}$$

$$\text{depth}(SM_A) \equiv \min\left\{ l \,\middle|\, \begin{array}{l} l = \text{length}\left(\text{path}_{\text{Composition}}(SM_S, SM_A)\right), \\ SM_S \in \text{StartVertices}(G_{\text{Composition}}) \end{array} \right\}$$

A merged Service Port in the Composite (Service) Structure (CompSt) (cf. Definition 24) can be interpreted as an edge connecting two selected Service Modes. In the same interpretation, a selected Service Mode becomes a vertex. In the current version of ACTAS, the selected Service Modes, i.e. the Request Modes, of Actor Service Offers (ASO) are seen as the starting vertices of the introduced Service Composition graph ($G_{Composition}$). The depth of a Service Composition of a selected Service Mode A (SM_A) is defined as the least number of merged Service Ports, which have to be used in CompSt, in order to come from an ASO to the selected Service Mode A. The depth of Service Composition of every ASO is zero in the current version of ACTAS.

Request and that an ASO is built from a Client Request of a Service Request. The algorithm of the Composition Process can use the field "depth", in order to sort the set of Open Ports in an ascending order. In this way, Open Ports with a smaller value of "depth" can be selected easier as COP, in order to extend the graph of composition (cf. Definition 27) ordered by its depth. The Composition Process could start a backtracking, when the "depth" value exceeds a certain threshold.

$$(\text{Sel-OpenSP}) \quad \frac{\text{COP can be selected from set of Open Ports}}{\text{take COP out of set of Open Ports,}}$$
$$\text{(COP stands for Currently selected Open Port)}$$

A more elaborated way to deal with the set of Open Ports can be achieved through the splitting up of the this set in sub sets for IN Ports, OUT Ports, and non-directed Open Ports. In these sets, only OUT Ports and non-directed Open Ports are Open Ports interesting for a selection as COP, since IN Ports only offer a service. Such a division of the set of Open Ports supports the rule that only then the OUT Ports of a selected Service Mode will be considered as Open Ports (i.e. added to the sub set of OUT Ports), when all IN Ports of this selected Service Mode have been merged. In this way, loops in the paths of the Service Composition can be avoided (proof in the evaluation chapter). A Composition Process can be seen as failed, when the set of IN Ports still contains elements, although the set of OUT Ports is empty. In this case, the Composition Process will start a backtracking. A Composition Process will fail completely, when the backtracking fails. In this section, it is assumed that the Composition Process is using the "depth" field for selection. It is implicit that only OUT Ports and non-directed Ports are selected, without mentioning their distinct sets any longer.

$$\text{(Match-OpenSP)} \quad \frac{\text{COP, OP} \in \text{Open Ports with (COP,OP) are principally compatible}}{\begin{array}{c}\text{take OP out of set of Open Ports and name it } \text{SP}_0 \\ \text{find Service Mode } \text{SM}_0 \text{ holding } \text{SP}_0 \\ \text{adapt depth value for all Open Ports of } \text{SM}_0 \text{ (without OP)} \\ \text{do for COP and } \text{SP}_0 \text{ rule (Add-Merged-SP)} \\ \text{do for new Merged-SP rule (Add-MeProperty)}\end{array}}$$

After the selection, of the COP, it should be first clarified, if there is not already an Open Port, which is principally compatible with the COP. This is tackled with rule (Match-OpenSP). If this rule does not find an Open Port OP, which is principally compatible with the COP, then the rule (Match-Discovery) for the call of the Service Discovery will be chosen. It builds a Trading Request (TRe) from the data of the COP, in order to do the search for a principal compatible Service Offer. The necessary information for the testing of the principal compatibility (cf. Definition 11) is the set of references to Compatibility Characteristics $(\text{CCh}^{\text{Ref}})^{\text{Set}}$ (this includes Reference Characteristics) as well as the Option-Slots of the Open Port. It is to mention that the TRe contains a complete environment description (TRe-Env) allowing an extended testing of compatibility and consistency by the agents receiving the TRe. These agents are the Facility Agents (FA) and the Trader Agents (TrA) (cf. chapter 8).

$$\text{(Match-Discovery)} \quad \frac{\text{COP shall be composed with a new SO}}{\begin{array}{c}\text{build TRe of OP} \\ \text{Receive set of Candidate Service Offers } \text{SO}_{\text{Cand}}^{\text{Set}}\end{array}}$$

The Option-Slots for the environment TRe-Env can be found in the entry of the Open Port in CompSt-plus as well as in the referenced Service Port. This reference is kept in the entry of the Open Port. It seems the easiest way, to fetch the Exchange Constraints and Value Constraints also from this referenced Service Port. Nevertheless, it can be of higher interest to get the Value and Exchange Constraints out of the central sets of the CompSt-plus data-structure, since these sets might already contain an extended number of constraints concerning the Service Properties of the TRe. The rule (Get-Value-Co) for the Value Constraints showed principally how to fetch these constraints from the central environment set of CompSt-plus. However, the translation should not take place, since the informal data structures of the S-Model are needed, in order to build the XML message of the Trading Request.

The result of the Trading Request should be a set of Candidate Service Offers. In the return message to the Trading Request, the agents might only send the current SOER or this SOER in combination with a reference to the principally compatible Service Mode. Then the CoA will build the Service Offer (SO) as explained in section 13.2.1 - Service Offer. The handling of a Candidate Service Offer (SO_{Cand}) is shown in rule (Sel-Cand). Since the handling of the principally compatible Open Port (OP) in rule (Match-OpenSP) is quite similar, the procedure for both rules is discussed in parallel in the next paragraphs.

$$\text{(Sel-Cand)}\ \frac{\text{candidate } SO_{Cand} \text{ left in } SO^{Set}_{Cand},\ COP}{\begin{array}{c}\text{find principal compatible Service Port } SP_C \text{ of } SO_{Cand}\\ \text{find Service Mode } SM_C \text{ holding } SP_C\\ \text{do the selection for } SM_C \text{ rule (Sel-SO-SM)}\\ \text{do for all SP of } SM_C \text{ without } SP_C \text{ rule (Add-Open-SP)}\\ \text{do for COP and } SP_C \text{ rule (Add-Merged-SP)}\\ \text{do for new Merged-SP rule (Add-MeProperty)}\end{array}}$$

Both rules select candidates, which are principally compatible Service Ports with the COP (SP_O and SP_C respectively). On the one hand, the rule (Match-OpenSP) takes its candidate OP out of the set of Open Ports and renames it to SP_O. On the other hand, rule (Sel-Cand) selects the candidate from a list of candidates received from a Trading Request. However, the principally compatible Service Port SP_C of the Candidate Service Offer SO_{Cand} in rule (Sel-Cand) has still to be determined on the base of the return message of the Trading Request. In the end, the principally compatible Service Port will lead to a detection of the principal compatible Service Mode (SM_C and SM_O), i.e. the Service Mode to which the SP_C or SP_O belongs respectively.

$$\text{(Sel-SO-SM)}\ \frac{\begin{array}{c}\left(\text{Va-Co}^{Set}, \text{Me-Co}^{Set}, \text{Ex-Co}^{Set}\right)_{\text{CompSt-plus}}, Object^{Set}_{\text{CompSt}}, \text{SO-Env}_{SO}, \text{SO-Env}_{SM},\\ \text{SCommonProperty}^{Set}, \text{SModeProperty}^{Set}, \text{Object}^{Set}_{SO+SM}\end{array}}{\begin{array}{c}\left((\text{Va-Co}^{Set})', \text{Me-Co}^{Set}, (\text{Ex-Co}^{Set})'\right)_{\text{CompSt-plus}}, \left(Object^{Set}_{\text{CompSt}}\right)'\\ \text{SOER}^{Ref}, \text{SO-SM}^{Ref}, \text{Res-Info}, ModeProperty^{Set}\end{array}}\ if$$

$$\begin{array}{c}\text{SO-SM}^{Ref} \text{ references selected SM of SO, Res-Info=None,}\\ \left(\text{Object}^{Set}_{\text{CompSt}}\right)'=\text{Object}^{Set}_{\text{CompSt}} \cup \text{Object}^{Set}_{SO+SM},\\ \left(\text{Va-Co}^{Set}\right)'= \text{Va-Co}^{Set} \cup \text{getVaCo}(\text{ SO-Env}_{SM} \cup \text{ SO-Env}_{SO}),\\ \left(\text{Ex-Co}^{Set}\right)'= \text{Ex-Co}^{Set} \cup \text{getExCo}(\text{ SO-Env}_{SM} \cup \text{ SO-Env}_{SO}),\\ \text{ModeProperty}^{Set}= \text{SModeProperty}^{Set} \cup \text{SCommonProperty}^{Set} \text{ with key } (\text{GCh}^{Ref}, \text{Name})\end{array}$$

In case of rule (Sel-Cand), this Service Mode (SM_C) becomes the next selected Service Mode through rule (Sel-SO-SM). In the subsequent action of rule (Sel-Cand), all open Service Ports of this selected Service Mode with the exception of the just found principally compatible Service Port SP_C are made to Open Ports with the rule (Add-Open-SP). The new Open Ports get a "depth" value, which is the increment of the one of COP. The rule (Sel-SO-SM) is quite similar to the rule (Sel-ASO-RM) (cf. page 139) due to the duality of the functions and data structures of SO and ASO (this duality was described in section 13.2). Therefore, only the field "Res-Info" shall be mentioned. For the support of the negotiation between CoA and FA, CompSt can contain data about the reservation (Res-Info). It is assumed, that no communication between CoA and FA in the context of reservation took place until the application of the rule (Sel-SO-SM). Thus, the value is set to "None". Further explanations to this field are in section 13.7 - Step 6: Post-Processing.

In the case of rule (Match-OpenSP), the selection of the Service Mode and the addition of the Open Ports are not necessary, since the Service Mode of SP_O, i.e. SM_O, was already selected. Nevertheless, the "depth" value of its left Open Ports will be updated to the incremented one of COP, when the incremented value of COP is less than the one of SP_O, i.e. the original "depth" value of the Open Ports of SM_O, because it means that a shorter path of composition (cf. Definition 27) leads via the edge of COP and SP_O to the selected Service Mode SM_O. In fact, the update of the "depth" values of Open Ports is a recursive act, since the selected Service Mode SM_O is already composed with at least one other Service/Request Mode (the reason, why it was selected in the first place). These composed Service Modes might have also Open Ports, which could become reachable through a shorter path of composition via the mentioned edge. Therefore, recursively the "depth" values of the Open Ports of the composed Service Modes are tested against the value of COP, whereupon the value of COP is incremented with every extension of its path. Thus, the recursion will surely stop, when the several times incremented value of COP is not any longer smaller than the values held in the Open Ports. The evaluation looks at situations of Service Composition, when this recursive update of the "depth" values of Open Ports can occur.

Finally, the found principally compatible Service Ports (SP_O in rule (Match-OpenSP) and SP_C in rule (Sel-Cand)) get "merged" with the COP in a merged Service Port through the rule (Add-Merged-SP). This "merge" is not complete, because the Merge Constraints will be still applied later (described in section 13.5 - Step 4: Checking of Merge Constraints). However, through the rule (Add-MeProperty), the Comparable Service Properties are associated and the fitting Merge Constraints built. This is done in such a way that later the Object Reference of the Merge Property Object will be correct.

$$\text{(Add-Merged-SP)}\frac{\text{Option-Slot}^{Set}_{COP}, \text{Option-Slot}^{Set}_{SP}, \text{COP}, \text{SP}}{\text{Cl-SP}^{\text{Ref}}, \text{Se-SP}^{\text{Ref}}, \text{Directed}}\ \ if$$

COP and SP are principally compatible Service Ports (cf. Definition 11),
"request" Option-Slot is set: then SP and COP must be Request Ports (B2C composition)

"direction" Option-Slot is set, COP is an OUT Port, SP is an IN Port:
then Directed=true, $\text{Cl-SP}^{\text{Ref}}$ gets SP^{Ref} of COP, $\text{Se-SP}^{\text{Ref}}$ points to SP

"direction" Option-Slot is set, COP is an IN Port, SP is an OUT Port:
then Directed=true, $\text{Se-SP}^{\text{Ref}}$ gets SP^{Ref} of COP, $\text{Cl-SP}^{\text{Ref}}$ points to SP,
otherwise: Directed=false, $\text{Cl-SP}^{\text{Ref}}$ gets SP^{Ref} of COP, $\text{Se-SP}^{\text{Ref}}$ points to SP

The rule (Add-Merged-SP) creates a merged Service Port (Merged-SP) in CompSt, i.e. an edge in the graph of composition as defined by Definition 27. It sets the references to the Client Port ($\text{Cl-SP}^{\text{Ref}}$) and the Service Port ($\text{Se-SP}^{\text{Ref}}$) as well as a flag (Directed), if the composition was a directed one. The premise of the application of the rule is the existence of a principally compatible Service/Request Port to the currently selected Open Port (COP) according the Definition 11. This means that both ports must have identical set of references of Compatibility Characteristics and matching Option-Slots. The set of Option-Slot of both entities are cited in the premise. The Service Port SP could be for instance the SP_O or SP_C of the rules discussed above.

ACTAS distinguishes between a composition with Request Ports and the one with Service Ports. Request Ports are a special kind of Service Ports demanding that all of its Compatibility Characteristics must be Request Characteristics. In this way, ACTAS supports a B2C and B2B-like composition (cf. S-Model and R-Model). The R-Model of ACTAS requires that Service Requests work only with Request Ports. Therefore, the services must define also Request Ports, in order to be visible for a Service Request (cf. section 10.3 - Description of compatibility). A Request Port is marked with the "request" Option-Slot. The rule (Add-Merged-SP) checks the fulfilment of the requirements implied with "request" Option-Slot.

The "direction" Option-Slot defines a Service Port as Out Port (also called Client Port) or as In Port (also called Server Port). Table 15 defines the matching of Option-Slots held by Service Ports. The matching constraints for the "direction" Option-Slot are also tested by the rule (Add-Merged-SP). This Option-Slot makes a composition to a directed one. Therefore, the flag "Directed" will be set, when the "direction" Option-Slot exists. The references to the Client Port and the Server Port are set accordingly. If the "direction" Option-Slot does not exist in both sets, the composition will be called a "non-directed" one. The distinction between Client and Service Port is not any longer possible. However, the rule establishes a certain order between the Service Ports with setting their references into the fields of the merged Service Port.

The rule (Add-MeProperty) builds the set of MeProperty entries in a merged Service Port (Merged-SP, MeProperty in CompSt, Definition 24) for all Service Properties declared in the referenced Compatibility Characteristics. The rule uses the reference of a Compatibility Characteristic and the name of a Service/Char Property $(CCh^{Ref}, Name)$ as key. Both "merged" Service Ports referenced in the merged Service Port ($Cl\text{-}SP^{Ref}$, $Se\text{-}SP^{Ref}$) have an identical set of referenced Compatibility Characteristics, because they were tested as principally compatible. For the setting of the descriptions of the Comparable Properties (cf. Definition 12), the rule (Add-MeProperty) takes over the order of the "merged" Service Ports established by the rule (Add-Merged-SP), i.e. each pair of Comparable Properties will have a description with a Client Object Reference ($Cl\text{-}Prop^{ORef}$) and a Server Object Reference ($Se\text{-}Prop^{ORef}$).

The setting of the object references implies the extension of the dictionary of Property Objects ($Object^{Set}_{CompSt}$) in the CompSt data structure (cf. Definition 24). The sets of Property Objects and the associated descriptions of their properties are taken from the referenced Service Ports in the merged Service Port: (1) $PortProperty^{Set}_{Cl}$, $Object^{Set}_{Cl}$ from the Service Port referenced with $Cl\text{-}SP^{Ref}$, and (2) $PortProperty^{Set}_{Se}$, $Object^{Set}_{Se}$ from the Service Port referenced with $Se\text{-}SP^{Ref}$.

$$(\text{Add-MeProperty})\ \frac{\begin{gathered}(\text{Va-Co}^{\text{Set}}, \text{Me-Co}^{\text{Set}},\ \text{Ex-Co}^{\text{Set}})_{\text{CompSt-plus}}, \mathit{Object}_{\text{CompSt}}^{\mathit{Set}},\\ \text{Cl-SP}^{\text{Ref}}, \text{Se-SP}^{\text{Ref}}, \text{Directed},\\ \text{PortProperty}_{Se}^{Set}, \text{Object}_{Se}^{\text{Set}}, \text{PortProperty}_{Cl}^{Set}, \text{Object}_{Cl}^{\text{Set}},\\ \text{Se-SP-Env}, \text{Cl-SP-Env}\end{gathered}}{\begin{gathered}\left((\text{Va-Co}^{\text{Set}})', (\text{Me-Co}^{\text{Set}})', (\text{Ex-Co}^{\text{Set}})'\right)_{\text{CompSt-plus}},\\ \left(\mathit{Object}_{\text{CompSt}}^{\mathit{Set}}\right)', \mathit{MeProperty}^{\mathit{Set}}\end{gathered}}\ \mathit{if}$$

$\text{Cl-SP}^{\text{Ref}}, \text{Se-SP}^{\text{Ref}}$, Directed comes from a Merged-SP ,
$\text{PortProperty}_{\text{Se}}^{\text{Set}}, \text{Object}_{\text{Se}}^{\text{Set}}$,Se-SP-Env comes from port referenced by $\text{Se-SP}^{\text{Ref}}$
$\text{PortProperty}_{\text{Cl}}^{\text{Set}}, \text{Object}_{\text{Cl}}^{\text{Set}}$,Cl-SP-Env comes from port referenced by $\text{Cl-SP}^{\text{Ref}}$,
$(\text{Va-Co}^{\text{Set}})' = \text{Va-Co}^{\text{Set}} \cup (\text{getVaCo(Se-SP-Env)} \cup \text{getVaCo(Cl-SP-Env)})$,
$(\text{Ex-Co}^{\text{Set}})' = \text{Ex-Co}^{\text{Set}} \cup (\text{getExCo(Se-SP-Env)} \cup \text{getExCo(Cl-SP-Env)})$,
$(\text{Me-Co}^{\text{Set}})' = \text{Me-Co}^{\text{Set}} \cup \text{Me-Co}_{New}^{Set}$,
$(\text{Object}_{\text{CompSt}}^{\text{Set}})' = \text{Object}_{\text{CompSt}}^{\text{Set}} \cup \text{Object}_{\text{Se}}^{\text{Set}} \cup \text{Object}_{\text{Cl}}^{\text{Set}}$,
For every $(\text{CCh}^{\text{Ref}}, \text{Name})$: create MeProperty with
Client Object Reference ($\text{Cl-Prop}^{\text{ORef}}$) from $\text{PortProperty}_{\text{Cl}}^{\text{Set}}$ and
Server Object Reference ($\text{Se-Prop}^{\text{ORef}}$) from $\text{PortProperty}_{\text{Se}}^{\text{Set}}$,
$\text{Me-Co}_{new} = \text{me-co}(\text{Mer-SP}^{Ref}, \text{CCh}^{\text{Ref}}, \text{Name}, \text{Directed}, \text{Cl-Prop}^{\text{ORef}}, \text{Se-Prop}^{\text{ORef}}, \text{Me-Prop}^{\text{ORef}})$

Option-Slots were interpreted by the rules and they do not become part of the environment information in the Common Part of the CompSt-plus data structure. However, the Value Constraints and the Exchange Constraints are taken over from both referenced Service Ports as described above in the context of other rules. A very important action of the rule (Add-MeProperty) is the creation of the Merge Constraints and their object references. In the following section 13.5 - Step 4: Checking of Merge Constraints, these Merger Constraints are applied, in order to get the "merged" Service Properties completely checked.

The Merge Constraint consists (cf. Definition 28) of the key (CCh^{Ref}, Name), the flag "Directed" from the merged Service Port (Merged-SP in CompSt) as well as object references to the Property Objects of the Client Property, Server Property, and the Merge Property (the order of the properties for a non-directed composition was also established through rule (Add-Merged-SP)). The Merge Constraint will check if the information in the Client Property can be matched with the information in the Server Property observing a directed relationship, when the flag "Directed" is set. In the case of a non-directed Service Composition, the values of the Service Properties of the merged Service Ports have to be adjusted. Further details will be in the next section.

One cycle of step 3 – Service Discovery and Principal Compatibility – concludes with the application of the Value Constraints on the Service Properties as explained in section 13.3.4. Value Constraints, which were already applied on Service Properties during the initialisation (step 2 of the Composition Process) or earlier cycles of step 3, are not anew applied. There will be further cycles of step 3 as long as Open Ports can be selected with rule (Sel-OpenSP). In general, the third step of the Composition Process will end successfully, when all Open Ports could be composed with other Service Ports. This section concludes with the continuation of the Example 17.

Example 17 showed the initialisation of the CompSt-plus data structure for a simple example of the telecommunication, in which two Service Clients liked to have a telecommunication. Example 18 extends this example with a possible progression of the Composition Process showed in Table 18 as a flow trace starting with the initialisation described in Example 17. In Fig. 38, the resulting composition graph (cf. Definition 27) including the Semantic Characteristics and some Option-Slots is illustrated.

The middle column of Table 18 shows the newly transferred information; sets of CompSt are portrayed with their keys in the third column. The elements, once mentioned in the middle column are not repeated. The example is based on the principal compatibility, i.e. the Service Properties are left out. For simplification, the identifications of the entities get symbolic names with consecutive indices like for instance SO_1 as an identification of the first discovered Service Offer of the example.

The rule (InitialiseStep2) leads to the selection of the two Request Modes of the two ASOs as the first selected Service Modes of CompSt with the keys: $\{(\text{CompSt}, \text{RM}_1), (\text{CompSt}, \text{RM}_2)\}$. The Request Ports of the ASOs become the first Open Ports $\{\text{OP}_1, \text{OP}_2\}$. The list of Open Ports is sorted with the "depth" value. This value was initialised with zero for the first selected Service Modes, the Request Modes of the ASOs.

Example 18 **Telecommunication** with Gateway **(continuation1)**

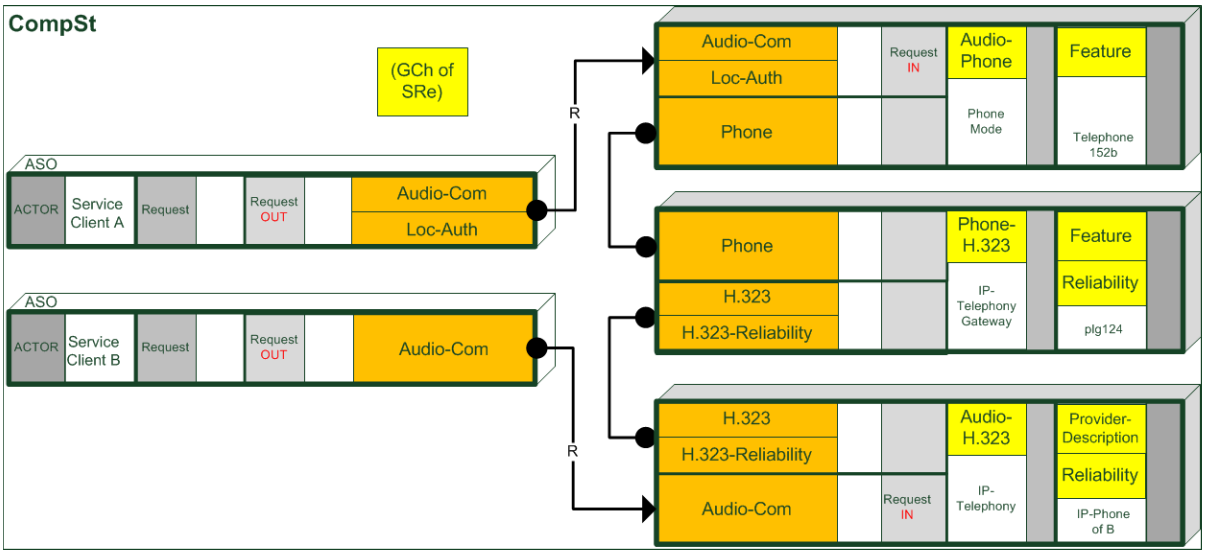


Fig. 38 - Technical Service shown with principal compatibility

Action/ Result	Transferred Information	Contents of sets
	Service Request leads to an initialisation as shown in Fig. 37 - Initialised Composite Structure …	

ACTAS - Composition Model (C-Model)

Action/ Result	Transferred Information	Contents of sets
Initialisation of CompSt-plus with ASO_1 and ASO_2	Selected-SM: $((\text{CompSt,RM}_1), \text{SRe}_1, (\text{ASO}_1,\text{RM}_1), \_, \_)$ Selected-SM: $((\text{CompSt,RM}_2), \text{SRe}_1, (\text{ASO}_2,\text{RM}_1), \_, \_)$ open-SP: $\begin{pmatrix} \text{OP}_1, (\text{ASO}_1,\text{RM}_1,\text{RP}_1), \\ [\text{reqest, direction(OUT)}], \\ 0, [\text{Audio-Com, Loc-Auth}] \end{pmatrix}$ open-SP: $\begin{pmatrix} \text{OP}_2, (\text{ASO}_2,\text{RM}_1,\text{RP}_1), \\ [\text{reqest, direction(OUT)}], \\ 0, [\text{Audio-Com}] \end{pmatrix}$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix} (\text{CompSt, RM}_1), \\ (\text{CompSt, RM}_2) \end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_1,\text{OP}_2\}$ $\text{Merged-SP}^{Set} = \{ \}$
Selection of OP_1 as COP, Building of TRe_1	trading-Re: $\begin{pmatrix} \text{TRe}_1, \text{All-Agents}, \\ [\text{Audio-Com, Loc-Auth}], \\ \text{TRe-Env}_1 \end{pmatrix}$ TRe-Env_1:(_,_,[reqest, direction(OUT)])	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix} (\text{CompSt, RM}_1), \\ (\text{CompSt, RM}_2) \end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_2\}$ $\text{Merged-SP}^{Set} = \{ \}$
Discovery of SO_1 and principal compatibility with Service/Request Port $(\text{SO}_1,\text{SM}_1, \text{SP}_1)$ Selection of $(\text{SO}_1, \text{SM}_1)$	SO-SP : $\begin{pmatrix} (\text{SO}_1,\text{SM}_1,\text{SP}_1), [\text{Audio-Com, Loc-Auth}], \_, \\ (\_,\_,[\text{request,direction(IN)}]) \end{pmatrix}$ Selected-SM: $\begin{pmatrix} (\text{CompSt},SM_1), SOER_1, (SO_1, \text{SM}_1), \\ \text{None}, \_, \_ \end{pmatrix}$ Merged-SP: $\begin{pmatrix} \text{Mer-SP}_1, (\text{ASO}_1,\text{RM}_1,\text{RP}_1), \\ (\text{SO}_1,\text{SM}_1,\text{SP}_1), \\ \text{YES, props(Audio-Com,Loc-Auth)} \end{pmatrix}$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix} (\text{CompSt, RM}_1), \\ (\text{CompSt, RM}_2), \\ (\text{CompSt, SM}_1) \end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_2\}$ $\text{Merged-SP}^{Set} = \{\text{Mer-SP}_1\}$
Building of new Open SP	open-SP:$(\text{OP}_3, (\text{SO}_1,\text{SM}_1,\text{SP}_2), [\ ], 1,[\text{Phone}])$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix} (\text{CompSt, RM}_1), \\ (\text{CompSt, RM}_2), \\ (\text{CompSt, SM}_1) \end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_2, \text{OP}_3\}$ $\text{Merged-SP}^{Set} = \{\text{Mer-SP}_1\}$

Action/ Result	Transferred Information	Contents of sets
Selection of OP$_2$ as COP, Building of TRe$_2$ Discovery of SO$_2$ and Selection Building of new Open SP	trading-Re:$(\text{TRe}_2, \text{All-Agents}, [\text{Audio-Com}], \text{TRe-Env}_2)$ SO-SP : $\begin{pmatrix}(\text{SO}_2,\text{SM}_1,\text{SP}_1), [\text{Audio-Com}], \_, \\ (\_,\_,[\text{request},\text{direction}(\text{IN})])\end{pmatrix}$ Selected-SM: $\begin{pmatrix}(\text{CompSt},SM_2), SOER_2, (SO_2, \text{SM}_1), \\ \text{None}, \_, \_\end{pmatrix}$ Merged-SP: $\begin{pmatrix}\text{Mer-SP}_2, (\text{ASO}_2,\text{RM}_1,\text{RP}_1), \\ (\text{SO}_2,\text{SM}_1,\text{SP}_1), \\ \text{YES}, \text{props}(\text{Audio-Com})\end{pmatrix}$ open-SP: $\begin{pmatrix}\text{OP}_4, (\text{SO}_2,\text{SM}_1,\text{SP}_2), [\ \ ], 1, \\ [\text{H.323}, \text{H.323-Reliability}]\end{pmatrix}$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix}(\text{CompSt}, \text{RM}_1), \\ (\text{CompSt}, \text{RM}_2), \\ (\text{CompSt}, \text{SM}_1), \\ (\text{CompSt}, \text{SM}_2)\end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_3, \text{OP}_4\}$ $\text{Merged-SP}^{Set} = \{\text{Mer-SP}_1, \text{Mer-SP}_2\}$
Selection of OP$_3$ as COP, Building of TRe$_3$ Discovery of SO$_3$ and Selection Building of new Open SP	trading-Re: $(\text{TRe}_3, \text{All-Agents}, [\text{Phone}], (\_,\_,[\ \ ]))$ SO-SP :$((SO_3, SM_1, SP_1), [\text{Phone}], \_, (\_,\_,[\ \ ]))$ Selected-SM: $\begin{pmatrix}(\text{CompSt},\text{SM}_3), SOER_3, (\text{SO}_3, \text{SM}_1), \\ \text{None}, \_, \_\end{pmatrix}$ Merged-SP: $\begin{pmatrix}\text{Mer-SP}_3, (\text{SO}_1,\text{SM}_1,\text{SP}_2), \\ (\text{SO}_3,\text{SM}_1,\text{SP}_1), \\ \text{NO}, [\text{Phone}]\end{pmatrix}$ open-SP: $\begin{pmatrix}\text{OP}_5, (\text{SO}_3,\text{SM}_1,\text{SP}_2), [\ \ ], 2, \\ [\text{H.323}, \text{H.323-Reliability}]\end{pmatrix}$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix}(\text{CompSt}, \text{RM}_1), \\ (\text{CompSt}, \text{RM}_2), \\ (\text{CompSt}, \text{SM}_1), \\ (\text{CompSt}, \text{SM}_2), \\ (\text{CompSt}, \text{SM}_3)\end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\text{OP}_4, \text{OP}_5\}$ $\text{Merged-SP}^{Set} = \begin{Bmatrix}\text{Mer-SP}_1, \text{Mer-SP}_2, \\ \text{Mer-SP}_3\end{Bmatrix}$
Selection of OP$_4$ as COP, Discovery of principal compatibility with OP_5	SO-SP : $\begin{pmatrix}(SO_3, SM_1, SP_2), \\ [\text{H.323}, \text{H.323-Reliability}], \\ (\_,\_,[\ \ ])\end{pmatrix}$ Merged-SP: $\begin{pmatrix}\text{Mer-SP}_4, (\text{SO}_2,\text{SM}_1,\text{SP}_2), \\ (\text{SO}_3,\text{SM}_1,\text{SP}_2), \\ \text{NO}, [\text{H.323}, \text{H.323-Reliability}]\end{pmatrix}$	$\text{Selected-SM}^{\text{Set}} = \begin{Bmatrix}(\text{CompSt}, \text{RM}_1), \\ (\text{CompSt}, \text{RM}_2), \\ (\text{CompSt}, \text{SM}_1), \\ (\text{CompSt}, \text{SM}_2), \\ (\text{CompSt}, \text{SM}_3)\end{Bmatrix}$ $\text{Open-SP}^{Set} = \{\ \ \}$ $\text{Merged-SP}^{Set} = \begin{Bmatrix}\text{Mer-SP}_1, \text{Mer-SP}_2, \\ \text{Mer-SP}_3, \text{Mer-SP}_4\end{Bmatrix}$

Table 20 - Flow Trace of Service Composition with CompSt

The Open Port OP_1 was selected as first COP (rule (Sel-OpenSP)). Since the rule (Match-OpenSP) was not applicable, a Trading Request is built and sent to "all agents" with rule (Match-Discovery) by the CoA. The environment of the Trading Request contains the "request" and the "direction" Option-Slots, which marks the COP as a Client Port of a B2C-like Service Composition. The two Request Characteristics request an audio communication through a communication facility, which is reachable and accessible. The Request Characteristic "Loc-Auth" ensures that the Service Offer understands the same under reachability and accessibility of facilities as the Service Request. The Merge Constraints applied on the Service Properties held by this Semantic Characteristic would clarify the matching further on. The application of Merge Constraints is covered in the next section.

The principal compatibility leads to the selection of Service Mode SM_1 of Service Offer SO_1. The General Characteristics belonging to the Common Part of the Service Offer will only become part of the selected Service Mode, when they do not also appear in the SM part of the Service Description (cf. Definition 8 and Definition 22). In the current version of ACTAS, a certain General Characteristic can only appear once in a selected Service Mode. The rule (Sel-Cand) selects the next selected Service Mode and creates the "merged" Service Port (Merged-SP) for the principal compatible Service Ports (COP and the Service Port SPC of the selected Service Mode (SO1, SM1) (cf. rule (Sel-Cand)). The non-composed Service Port of the selected Service Mode becomes the new Open Port OP_3. The "depth" value of this new Open Port is incremented and its Option-Slots declare it to a Service Port for a non-directed, B2B-like Service Composition, since neither a "request" Option-Slot nor a "direction" Option-Slot is set.

The cycle is repeated for the next selected Open Port OP_2 as COP, leading to the selection of the principally compatible Service Mode (SO_2, SM_1) and a merged Service Port with the COP and the principally compatible Service Port of (SO_2, SM_1). At the end of this cycle, a new Open Port OP_4 is created. However, the last selected Service Mode (SO_2, SM_1) offers a communication facility, which does not use the same standards or network as the selected Service Mode (SO_1, SM_1). Thus, the Open Ports OP_3 and OP_4 are not principally compatible.

Therefore, rule (Match-Discovery) has to be applied again in the next cycle, in order to discover a principally compatible Service Port for the Open Port OP_3, as the next COP. In the example, it results in the discovery of a gateway service and the selection of the Service Mode (SO_3, SM_1). This cycle concludes with the creation of a merged Service Port composing the COP and the principally compatible Service Port of the selected Service Mode (SO_3, SM_1) as well as building of the new Open Port OP_5. Since OP_5 belongs to the selected Service Mode of the gateway service, its "depth" value was incremented and equals two.

Due to the lower "depth" value, the Open Port OP_4 is selected as next COP in the following cycle. The rule (Match-OpenSP) can be applied now, since OP_5 turns out as principally compatible offering a non-directed Service Composition with the same standard "H.323" and a function "H.323-Reliability" for the testing of the reliability of a connection base on this standard. The selection of the gateway service and the addition of its Open Ports are not necessary, because this happened already for the gateway service in the last cycle. However, the merged Service Port for COP and the Service Port referenced in the entry of OP_5 has still to be created, in order to complete the composition graph.

As the set of Open Ports is empty, the Composition Process in the example comes to a successful ending. The next steps are the checking of the Merge Constraints and of the Exchange Constraints. The actions for this testing are described more in detail in the next sections. The algorithm of the Composition Agent (CoA) for the composition of telecommunication could be improved in comparison to the one introduced in this section about the step 3 of the Composition Process. It is obviously a goal of the algorithm to achieve a connected graph as early as possible. The Composition Process starts with as many sub-graphs as ASOs exist. Therefore, the selection of candidates in rule (Sel-Cand) should prefer Service Offer Candidates, which connect as many sub-graphs as possible.

13.5 Step 4: Checking of Merge Constraints

The Merge Constraint (cf. Definition 16) completes the "merge" of the principally compatible Service Ports, which were discovered and saved as merged Service Ports in the field Merged-SP of the Composite Structure CompSt in the previous step of the Composition Process. The Merge Constraints perform the matching and mediation of the values of each pair of Comparable Properties of this Merged-SP (cf. Definition 12 and Fig. 32 - Me-Constraints for directed and non-directed composition). The simple definition of the principal compatibility, which just demands the same referenced Compatibility Characteristics, i.e. the same semantic contexts for the description of compatibility, led to the building of pairs of Service Properties (Comparable Properties), which are not only semantically comparable, but can be matched and mediated with proven, semantically fitting, and reliable algorithms, accessible through the Merge Property Class. Through his choice of a reference to a certain Compatibility Characteristic in the Service Description (cf. S-Model), the Service Designer selected implicitly the approved algorithms retrieved by the Merge Property Classes. The Service Requester decides for the same algorithms through the use of a reference to the same Compatibility Characteristic in his Service Request (cf. R-Model).

For directed Service Compositions (the flag "Directed" is set), the Merge Constraint will check if the information in the Client Property can be matched with the information in the Server Property observing a directed relationship. Examples for directed Service Composition will be discussed in the evaluation. In the case of a non-directed Service Composition, the values of the Service Properties of the merged Service Ports have to be adjusted. For instance the "Speed" Service Properties of the Compatibility Characteristics "Phone" in Fig. 32 could hold different ranges of acceptable line speeds. The Merge Constraint will select an adjusted common range according its policies using an appropriate, approved algorithm.

Definition 28. Merge Constraint (Me-Co) in C-Model

$$\text{MergeConstraint} \equiv \text{me-co}\begin{pmatrix} \text{Mer-SP}^{Ref}, \text{CCh}^{\text{Ref}}, \\ \text{PropName}, \text{Directed}, \text{Cl-Prop}^{\text{ORef}}, \text{Se-Prop}^{\text{ORef}}, \ \text{Me-Prop}^{\text{ORef}} \end{pmatrix}$$

Meaning:

- Reference of the Merged-SP of the Comparable Properties
- Reference to the common Compatibility Characteristic (CCh^{Ref})
- Name of the Comparable Property (PropName) (cf. Definition 12)
- $\text{MergePropClass}^{\text{Ref}}$ can be found with the information (CCh^{Ref}, PropName). This Merge Property Class is used, in order to construct the Merge Property Object referenced through $\text{Me-Prop}^{\text{ORef}}$.
- References to Char Property Objects of comparable Service Properties ($\text{Cl-Prop}^{\text{ORef}}$, $\text{Se-Prop}^{\text{ORef}}$)
- BOOLEAN "Directed" distinguishes whether the Merge Constraint is applied for a directed or non-directed Service Composition. For a directed Service Composition, the Service Properties of the client and of the server side are hold in $\text{Cl-Prop}^{\text{ORef}}$ or $\text{Se-Prop}^{\text{ORef}}$, respectively.

The Merge Constraints for a Service Composition exist only in the C-Model. They are built through an action of rule (Add-MeProperty) explained in the previous step of the Composition Process (cf. section 13.4 - Step 3: Service Discovery and Principal Compatibility). The Merge Property Object, constructed through the application of the Merge Constraint, is saved in the object dictionary ($\text{object}^{\text{Set}}$) of the Composite Structure (CompSt, cf. Definition 24). The Merge Constraint (Definition 28) keeps an object reference in $\text{Me-Prop}^{\text{ORef}}$. The Composite Structure (CompSt, Definition 24) also holds an object reference to the Merge Property Object in a MeProperty entry of the Merge Property Description of the merged Service Port (Merged-SP).

The Merge Constraint keeps further object references to the Property Objects to the pair of Comparable Properties. In the Composite Structure (CompSt, cf. Definition 24), Comparable Properties are held in MeProperty entries of the Merge Property Description of a merged Service Port (Merged-SP). Comparable Properties have the same name (PropName) in the same semantic context of a Compatibility Characteristic (referenced through CCh^{Ref}).

ACTAS demands the implementation of at least three methods of the implantation instances of a Merge Property Class accessible through the handle of a Merge Property Object: (1) the "merge" method, (2) the "export" method, and (3) the "import" method. These methods are semantically defined through the rules (Merge), (Export), and (Import), respectively, and discussed in this section. The method "merge" implements the application of the Merge Constraints and the construction of the Merge Property Object. The methods "export" and "import" are used for the evaluation of Exchange Constraints, in order to realize the access of the Service Properties linked with the Exchange Name (ExName) through the Exchange

Properties term (cf. Definition 18). The subsequent section looks at the application of Exchange Constraints in the C-Model.

$$(\text{Merge})\frac{op\begin{pmatrix}MergePropertyClass, merge, [Directed],\ [ClO, SeO],\\ [ClO', SeO', MergeObject]\end{pmatrix}}{MergeObject, ClO', SeO'}\ if$$

$$\exists \text{me-co}\left(\text{Mer-SP}^{Ref}, \text{CCh}^{\text{Ref}}, \text{PropName, Directed, Cl-Prop}^{\text{ORef}}, \text{Se-Prop}^{\text{ORef}},\ \text{Me-Prop}^{\text{ORef}}\right)$$
$$\in \text{MergeConstraints,}$$

MergePropertyClass was referenced through (CCh^{Ref}, PropName) and used for the construction of MergeObject,
MergeObject, ClO, ClO', SeO, SeO' are Property Objects,

The Merge Constraint is applied with the method “merge” through the op/5 predicate, which is shown in the rule (Merge). The predicate takes as parameter the “Directed” flag. The Property Objects of the Client Property and of the Server Property are the input objects (The order of these fields for non-directed Service Composition is determined in rule (Add-MeProperty).). The newly created output objects are Property Objects for the client and server side as well as the Merge Property Object discussed above.

$$(\text{Get-Merge-Co})\frac{\text{Me-Co}^{\text{Set}}, Property^{Ref}}{\text{me-co}(\text{Property}^{\text{Ref}}, \text{Directed, Cl-Prop}^{\text{ORef}}, \text{Se-Prop}^{\text{ORef}},\ \text{Me-Prop}^{\text{ORef}})}\ if$$

$\text{Property}^{\text{Ref}}$describes the Merged-SP, the CCh and the Name of the Comparable Properties

Finally, it must be possible to select the Merge Constraints from the environment set $\text{Me-Co}^{\text{Set}}$ of the Composite Structure CompSt (cf. Definition 24), in order to apply the rule (Merge), i.e. the predicate op/5 for the “merge” method. The rule (Get-Merge-Co) allows this selection through a reference of the Comparable Property, which includes its merged Service Port (Merged-SP). A Merge Constraint is applied only once and then deleted from the environment set. The Merge Property Object keeps accessible through the object reference in its MeProperty entry of the merged Service Port (Merged-SP) in the Composite Structure (CompSt).

A Merge Property Class and its Merge Property Object, constructed through the application of the Merge Constraint (cf. rule (Merge)), offer at least the three demanded methods (“merge”, “import”, and “export”). They can offer additional methods for instance for getting information about the merge results using the view on the Merge Property as defined in Definition 17. However, latest for the realizing of the so-called “merged view” on the Merge Property Object, it is a good idea to implement its implementation instance statefully, i.e. for the keeping of information. ACTAS does not demand the keeping of information inside of the implementation instance of a Merge Property Object. It is up to the provider of the implementation of the Merge Property Class, how he keeps the information in the implementation instances of the Property

Objects of the client and server side as well as the Merge Property Object. Therefore, the methods of the Merge Property Class get as input objects always the Property Objects of the client and server side, too. The "import" and "export" methods are explained in the next section 13.6 - Step 5: Checking of Exchange Constraints.

13.6 Step 5: Checking of Exchange Constraints

In section 10.4.3, Exchange Constraints were introduced in the S-Model as constraints establishing relationships between Service Properties of different Semantic Characteristics and parts of the Service Description. The Example 15 and the discussion of Case Study 2: Distribute Feature Composition (DFC) show applications of Exchange Constraints. The chapter illustrated the possibility to introduce with the Common Part of the Service Request Exchange Constraints, which can generally involve any Service Properties of the Composite Structure CompSt. The Example 15 portrayed the "building block" concept with a General Characteristic, which interlinked two Compatibility Characteristics through an Exchange Constraint. The Service Designer could adapt the Exchange Constraints to the Service Properties of his Service Description with the "exchangeProperties" Option-Slot (cf. Table 14 - Option-Slots of Service Modes and Common Part of Service Descriptions).

$$(\text{Get-Exchange-Co})\frac{\text{Ex-Co}^{\text{Set}}, \text{Ex-Co}^{Ref}}{\text{ex-co}(\text{Ex-Co}^{ID}, \text{ExchangeElements}, \text{ExchangeClause})}$$

In the fifth step of the general Composition Process described in this chapter, the Exchange Constraints are selected from the set of Exchange Constrains ($\text{Ex-Co}^{\text{Set}}$) in the environment description of the Composite Structure (CompSt), in order to be applied. The rule (Get-Exchange-Co)selects Exchange Constraints from this set of Exchange Constraints using as reference (Ex-Co^{Ref}) the identifications of the Exchange Constraints. The Definition 18 introduced the Exchange Clause consisting of a premise (a conjunction of left hand clauses (lh-clause)) and an entailment (a conjunction of right hand clauses (rh-clause)). The entailment of an Exchange Constraint will be only checked, when the premise is fulfilled. The application of an Exchange Constraint will be only named successful, when the clauses of the right hand side could be satisfied. Exchange Constraints will only be applied once, i.e. an Exchange Constraint is marked as fulfilled, when it was applied successfully.

This section discusses the application of an Exchange Constraint in the C-Model as shown in Definition 29. The Exchange Constraint uses its own Property Classes (cf. section 9.2 - S-Model: Property Classes), which are called Exchange Property Classes. The used ones are enumerated in the term "Exchange Classes". The methods of these Exchange Property Classes are not necessary statefull, i.e. ACTAS does not demand the storing of information. Thus the information for the Exchange Constraints has to be "borrowed" from the information of the Composite Structure (CompSt). After the application of the Exchange Constraint, this

Definition 29. Clause of Exchange Constraint (Ex-Co) in C-Model

$$\text{ExchangeConstraint} \equiv \text{ex-co}(\text{Ex-Co}^{ID}, \text{ExchangeElements}, \text{ExchangeClause})$$

ExchangeElements, ExchangeProperties, and ExchangeClasses like in Definition 18

$$\text{ExchangeClause} \equiv \begin{pmatrix} [\text{lh-clause}_1,\cdots, \text{lh-clause}_n] \Rightarrow \\ [\text{rh-clause}_1,\cdots, \text{rh-clause}_m] \end{pmatrix}$$

$$\text{lh-clause} \equiv \top \mid \text{opclause} \mid \text{testclause}$$
$$\text{rh-clause} \equiv \top \mid \text{opclause} \mid \text{testclause}$$
$$\text{opclause} \equiv$$
$$\text{op}\begin{pmatrix} \text{MethodObject, method,}[\text{Parameter}_1,\cdots,\text{Parameter}_o], [\text{ExObject}_1,\cdots,\text{ExObject}_p], \\ [ExObject_1^{new},\cdots, ExObject_o^{new}] \end{pmatrix}$$
$$\text{testclause} \equiv \text{test}\begin{pmatrix} \text{MethodObject, method,}[\text{Parameter}_1,\cdots,\text{Parameter}_l], \\ [\text{ExObject}_1,\cdots,\text{ExObject}_k] \end{pmatrix}$$

Meaning:

- $n, m, o, p, k \in \mathbb{N}$,
- $\text{Property}^{\text{Ref}}$ is a reference to the Service Property (cf. Definition 15) declared with the Char Property Class of the Char Property Objects PropObject_k, PropObject_{k+1}, respectively.
- ExObject_k is a Property Object, which granted all constraints coming from the $clause_1$ to $clause_{k-1}$ (cf. rule (ExOp))
- The ValueClause contains a list of methods calls with op/5 (i.e. semantically a conjunction of the resulting constraints is realised. ($\text{clause}_1 \wedge \cdots \wedge \text{clause}_m$)).
- In a successful case, a PropObject_{m+1} is generated.

information has to be incorporated into the information of the Composite Structure. Especially the Merge Constraints shall still be fulfilled for the new information. Therefore the discussion starts with the "export" and "import" of the Property Objects used in the Exchange Constraint. Then it has a closer look at the application itself and ends with the continuation of the examples mentioned above.

13.6.1 The access of information for the Exchange Constraint

The Exchange Clause contains Exchange Names for the descriptions of the input and output objects of the methods. The use of Exchange Names instead of concrete references of Service Properties allow a flexible and adaptive way of applying Exchange Constraints, since the Exchange Names can be linked with Service Properties as needed in a given semantic context. Due to this indirect way of accessing the Service Properties, General Characteristics can be introduced as "building blocks" holding Exchange Constraints, which can be adapted in the Service Description accordingly (cf. Example 15).

The Exchange Clause and the other terms of the Exchange Constraint can be found again in Definition 29 describing the Exchange Constraint as it is applied in the C-Model. However, the references of Service Properties linked with the Exchange Names have to be translated, in order to get Property Objects as handles to their implementation instances. Additionally, the method calls have to be wrapped in the predicates op/5 and test/4. The latter predicate tests the fulfilment of constraints at the time point of its application. However, it does not create new output objects, which preserved the resulting internal constraints (cf. section 13.1 - The Property Objects/Classes in the C-Model). Thus, the test/4 does not support the monotony of the application of constraints, but can be applied in order to test constraints, which shall not persist.

$$(\text{Apply-Ex-Co})\ \frac{\text{ex-co}(\text{Ex-Co}^{ID},\text{ExchangeElements, ExchangeClause})}{\text{construct Exchange Property Objects with ExchangeClasses}}\quad \text{if}$$

Borrow Exchange Name Objects with the term ExchangeProperties
Apply-ExchangeClause
Return ExchangeName Objects

ExchangeElements, ExchangeProperties, and
ExchangeClasses like in Definition 18

The Exchange Properties term (“ExchangeProperties”), which is the first part of the term “ExchangeElements” in the definition of an Exchange Constraint in the S-Model and R-Model, enumerates the affected Service Properties through references according Definition 15 and links them with the Exchange Names (ExNames) of the Exchange Constraint. The references of the Properties of the Exchange Constraint use the views on the Merge Property Object (cf. Definition 17), in order to link a Service Property with the Exchange Names (ExNames) (cf. Definition 18 and Example 15). In the C-Model, these links are used for the “borrow” of Property Objects (cf. rule (Export)) for the Exchange Names appearing in the Exchange Constraint. This means that the Exchange Constraint are working with their own implementation instances, in order to fit in the declarative environment as previously explained (cf. section 13.1).

Code 6 - Application and Translation of Exchange Constraints - shows parts of the actions necessary for the translation of the Exchange Constraints, given in the S-Model and R-Model, into the declarative environment of the C-Model. After a successful application of the Exchange Constraint, the Property Objects associated with the Exchange Names have to be returned to the information in the Composite (Service) Structure CompSt (cf. 13.6.2 - A closer look at the application of an Exchange Constraint). The access and the return of Property Objects are discussed in the next paragraphs.

ACTAS creates new Property Objects for the Exchange Names. These new Property Objects are clones of the ones, which provide the handles to the implementation instances of the referenced Service Properties and which are kept in the object dictionary ($\text{Object}^{\text{Set}}$ in CompSt). The cloning of Property Objects is necessary, in order to support the backtracking algorithm of the declarative environment as explained in section 13.1 - The Property Objects/Classes in the C-Model. If the referenced Service Property is part of a General Characteristic, it can directly be cloned and used in the Exchange Constraint. After the successful application of the Exchange

Constraint, the new Property Object of the Exchange Name has simply to replace the original Property Object of the Service Property in the object dictionary ($\text{Object}^{\text{Set}}$ in CompSt).

$$(\text{Export})\frac{\text{ExO}^{\text{Set}}, \text{op}\begin{pmatrix}\text{MerO, export, [View], [MeO,ClO,SeO],}\\ \text{[ExObject]}\end{pmatrix}}{\text{ExO'}^{\text{Set}}, \text{ExObject}} \text{ if}$$

$$\begin{gathered}\exists(\text{Property}^{\text{Ref}}, \text{View, ExName}) \in \text{ExchangeProperties},\\ \text{Property}^{\text{Ref}} \text{ references Merge Property},\\ \text{MeO, ClO, SeO, ExObject are Property Objects},\\ \text{ExO}'^{\text{Set}} \leftarrow \text{ExO}^{\text{Set}} \cup (\text{ExName, ExObject})\end{gathered}$$

On the one hand, it has to be clarified how to “export” a Property Object from the Merge Property Object, in order to have a Property Object for the Exchange Name, which just represents a certain view on the Merge Property Object. On the other hand, such a Property Object with a certain view on the Merge Property Object has to be “imported” again into the Merge Property Object after the successful application of the Exchange Constraint, in order to see if it still fulfils the Merge Constraints. The demanded methods “export” and “import” of a Merge Property Class/Object, mentioned in the previous section about Merge Constraints, should just provide these functions. The demanded semantic of these methods is shown by the rules (Export) and (Import).

The rule (Export) extends the object dictionary (ExO^{Set}), which holds Property Objects of the Exchange Names, which are later used in the application of the Exchange Constraint. This dictionary is indexed with the Exchange Names (ExNames), which are stated in the entries of the term “ExchangeProperties” of the Exchange Constraints. Through these entries, the Exchange Names are linked with referenced Service Properties. The link will provide a view on the Merge Property Object (cf. Definition 17), when the referenced Service Property is inside of a merged Service Port (Merged-SP in CompSt). The rule is applied for every entry of the term “ExchangeProperties” concerning a Service Property of a Merged-SP. It creates a new Property Object of the referenced “merged” Service Property with observing the view on its Merge Service Object. For this purpose, predicate op/5 applies the method “export” of the Merge Property Object MerO (The Merge Property Object was constructed through the Merge Constraint as explained in the previous section). The “export” method gets the parameter “View” (cf. Definition 17). The input objects of op/5 are the Merge Property Object MerO and the Char Property Objects of the client and server side (ClO and SeO). The output object is a new Char Property Object, which gives access to an implementation instance that holds information of the Merge Property Object in the wished view. This new Char Property Object becomes an element of the object dictionary ExO^{Set} with an entry, which associates the Exchange Name (ExName) mentioned in the key with it.

As discussed in the previous section, the implementation instances accessed through the Merge Property Object should be implemented statefully, in order to hold information at least of the “merged” view. However, ACTAS does not demand this statefullness, since it might not always be of sense to keep this “merged” view. Therefore, ACTAS provides all three Property Objects (MeO, ClO, and SeO) for the “export” method as input objects, in order to allow more freedom for the implementation of the provision of a new Property Object with the right view on the Merge Property Object.

$$\text{(Import)}\ \frac{\text{ExO}^{\text{Set}},\ \text{op}\begin{pmatrix}\text{MerO, import, [View], [MeO,ClO,SeO, ExObject],}\\ \text{[MeO}^{'}\text{, ClO}^{'}\text{, SeO}^{'}\text{]}\end{pmatrix}}{\text{ExO}'^{\text{Set}}\text{, MeO}'\text{, ClO}'\text{, SeO}'}\ \text{if}$$

$$\begin{gathered}\exists(\text{Property}^{\text{Ref}}\text{, View, ExName}) \in \text{ExchangeProperties,}\\ \text{Property}^{\text{Ref}}\ \text{references Merge Property,}\\ \text{information of ExObject can be imported with its view,}\\ \text{MeO, MeO}'\text{, ClO, ClO}'\text{, SeO, SeO}'\text{, ExObject are Property Objects,}\\ \text{ExO}'^{\text{Set}} \leftarrow \text{ExO}^{\text{Set}} \setminus (\text{ExName, ExObject})\end{gathered}$$

After the application of the Exchange Constraint a dictionary of new Property Objects associated with the Exchange Names is formed. The information accessed through these Property Objects has to be integrated or “imported” into the information of the Composite Structure (CompSt). More precisely, the information of a Property Object of an Exchange Name, which is linked with a reference to a Merge Property through an entry in the term “ExchangeProperties”, has to be “imported” into the information of its referenced Merge Property. This is done through the method “import” of the Merge Property Object, which is semantically described through rule (Import). The rule applies the op/5 predicate, in order to call the “import” method of the Merge Property Object (MeO). The input objects are the Merge Property Object as well as the Property Objects of the client and server side (ClO and SeO). Additionally, the Char Property Object associated with the Exchange Name (ExObject) is an input object. The ExObject is only imported, when its information representing a certain view on the Merge Property is not in conflict with the internal Merge Constraints. For instance, when the information presents a client side view on the Merge Property, the information will only accepted and imported as the new client view, when it still matches with the held information of the server side. The resulting output objects are the new Merge Property Object MeO’ as well as the new Char Property Objects ClO’ and SeO’. These new Property Objects replace the entries in the object dictionary of CompSt.

13.6.2 A closer look at the application of an Exchange Constraint

The Code 6 - Application and Translation of Exchange Constraints – is discussed in this sub-section. The main predicate of the code example is applyExchangeConstraint/1, which takes as parameter just the Exchange Constraint, which should be applied. It differentiates the Exchange Constraint directly into the terms “ExchangeProperties”, “ExchangeClasses”, and the “ExchangeClause”, which correspond to the terms of Definition 29.

The next predicate getExchangePropertiesVarList/2 is a technical predicate. It just extends every entry of the "ExchangeProperties" list with a logical variable, in order to have an easier translation of the clauses.

The predicate getObjectsOfMethods/4 traverses the methods of the clauses in the Exchange Clause (term "ExchangeClause"), in order to find or create the Property Objects giving access to the implementation instances of the Property Classes of the methods mentioned in the clauses. The term "ExchangeClasses" delivers the Exchange Property Classes, which are used for the creation of correlating Exchange Property Objects. As discussed in the context of Exchange Constraints in the S-Model, some test methods can have their origin in the Char Property Classes of the referred Service Properties in the term "ExchangeProperties". Therefore this term is the third input parameter of the predicate getObjectsOfMethods/4. The result is an object dictionary "ClassObjectList" with Exchange/Char Property Objects, which will be later used for the call of the methods.

The unification "ExchangeClause = (LhClauses, RhClauses)" just differentiates the Exchange Clause in its premise and entailment. The premise and the entailment are translated separately with two calls of the predicate translateExchangeClause/5. The predicate takes as first input parameter the object dictionary created in line 4 with predicate getObjectsOfMethods/4. The second parameter list is the list of "ExchangeProperties" extended with variables of line 3. A list of clauses as third parameter concludes the input. The idea of the translation is an adaptation of an object dictionary holding of (Char) Property Objects associated with the Exchange Names after an application of a clause of the Exchange Clause. The resulting object dictionary is the first output parameter of translateExchangeClause/5. The second one is the list of op/5 and test/4 calls. Each call is wrapped with object dictionaries holding the Property Objects of the Exchange Names before and after the call.

At this point, the logical variables, which extended the entries of the "ExchangeProperties" term, come handy, since they can be unified with the input and output objects respectively of the op/5 call. Then later in the application, the logical variables of the first object dictionary get simply unified with the Property Objects associated with the Exchange Names. This happens with predicate borrowAllObjects/1. This predicate also uses the "export" method of the Merge Property Object as explained in the previous sub-section. The predicate returnAllObjects/1 uses the "import" method of the Merge Object, in order to incorporate the resulting Property Objects into the data structure of CompSt.

At first, the application of the Exchange Constraint tested the premise. When the premise is true, a cut symbol "!" is given (line 13). The cut symbol cuts of the backtracking, i.e. the entailment must be fulfilled otherwise the predicate applyExchangeClause/3 and therefore the predicate applyExchangeConstraint/1 will fail. However, it has to be said that the shown code is a coarse simplification. It is not illustrated, what happens with Exchange Constraints, of which the premise is not fulfilled. Therefore, some remarks in this direction are added in the next paragraph.

An Exchange Constraint "fires", when the premise is satisfied. In this case, the entailment must be also fulfilled. A "fired" Exchange Constraint is taken out of the set of Exchange Constraints in the CompSt-plus data structure. This means that an Exchange Constraint is applied only once. The application of Exchange Constraints ends, when the set of Exchange Constraints is empty or no Exchange Constraints are left with a fulfilled premise. Until then the Exchange Constraints in the set will be tested repeatedly if it contains an Exchange Constraint, which can be "fired". After the application of an Exchange Constraint, the premise of another one might have become satisfiable. This leads to a general remark about the application of Merge Constraints and Exchange Constraints given in the subsequent section.

13.6.3 General Remark to the Application of Constraints

ACTAS is an open framework environment, i.e. permanently new Semantic Characteristics and Property Classes can be published and existing entities can be adapted, because the Semantic Characteristics and the Property Classes of the ontological repositories are only referenced. This openness of ACTAS allows an improvement of the algorithms used in the constraints. However, the Semantic Descriptions of the entities and the purpose of the Char Properties should be kept. In this way, the Semantic Characteristics can be used as commonly agreed "building blocks" in a reliable way. A Service Designer has not to use all Char Properties in his Service Description through the application of constraints (Value Constraints, Merge Constraints, and Exchange Constraints). In this sense, not every Char Property becomes necessarily a Service Property used for the Service Description by the Service Provider. The Semantic Characteristics could define Exchange Constraints, which would check, if the constraints between the diverse Char Properties were fulfilled.

The Merge Constraints and the Exchange Constraints have to deal with Service/Char Properties, which were not initialised. Therefore, every Char Property Class should offer a method testing the initialisation state of a Service Property. On the one hand, Merge Constraints will declare non-initialised Comparable Properties always as fulfilling the Merge Constraints. On the other hand, Merge Constraints, in which only one Service Property is not initialised, will have to decide in their context, whether they go ahead with the merge. However, the methods of the Merge Property Classes should deny any export necessary for the "borrowing" of objects for the Exchange Names during the application of Exchange Constraints. The Exchange Constraints themselves can test the state of initialisation in their premise. The premise of an Exchange Constraint should ensure that the application of the Exchange Constraint is most useful, because an Exchange Constraint "fires" only once, i.e. the information in the "borrowed" Property Objects should carry ideally as much information as possible, in order to make the most of the Exchange Constraint. In the end, ACTAS is not a programming environment. ACTAS does not claim to exclude all non-matching Service Candidates, but it helps to discover and to compose Service Candidates involving freely several aspects of services.

13.7 Step 6: Post-Processing

The System Environment of ACTAS describes the Facility Agent (FA) as responsible for the Service and Resource Management. Through the publication of SOERs, fitting to the general Service Descriptions done through Service Templates (ST), the FA is supposed to adapt its Service Descriptions to the currently available resources. In principle, each reservation or consumption of a service will lead to a changed situation of its resources, i.e. it will lead to a reaction of the FA. The FA might publish a new SOER declaring the previous ones as invalid.

A selected Service Mode in the Composite Service Structure (CompSt) comes from a Service Offer (SO), which is built on the base of a ST and a SOER. This information is kept in CompSt. However, in other Service Discovery approaches, it can happen that a selected Service Offer is not any longer valid, when the Composite Service shall be deployed. ACTAS supports a validity checking of the selected Service Mode, i.e. the FA can be asked whether the SOER is still valid. For the support of the negotiation between CoA and FA, CompSt can contain data about the reservation in the field "Res-Info" (cf. Definition 24). In this way, the CoA can inform the FA about a selection. Later on, when this selection turned out as a good one, the CoA could ask for a reservation of the Service Offer based on a certain SOER. In this negotiation, the found information in the Composition Process, held in the Service Property Objects of the Composite Structure, can be used for a more specific resource reservation. In future, with an elaborated System Environment of ACTAS existing, the Resource and Reservation Management of the Facility Agents will become content of further research.

In the covering of the models, their entities were introduced through examples discussing the Technical Services, in order to include non-directed Service Composition. Starting with a Service Template in Example 9, the design of Service Descriptions with Semantic Characteristics as "building blocks" was sketched in Example 15. The Example 17 started the description of the Composition Process for an example showing the use of a gateway in telecommunication. The Composition Process was continued in Example 18. In Example 19, the resulting Composite Structure is portrayed. The example is covered anew in chapter 15, in the context of Case Study 1: Technical Services with translation.

```
applyExchangeConstraint(ex-co(_,(properties(ExchangeProperties),
                  classes(ExchangeClasses)), ExchangeClause)) :-
      getExchangePropertiesVarList(ExchangeProperties, ExchangePropertiesVarIn),
      getObjectsOfMethods(ExchangeClause, ExchangeProperties, ExchangeClasses,
                  ClassObjectList),
      ExchangeClause = (LhClauses, RhClauses),
      translateExchangeClause(ClassObjectList, ExchangePropertiesVarIn, LhClauses,
                  ExchangePropertiesVarTmp, LhOpSet),
      translateExchangeClause(ClassObjectList, ExchangePropertiesVarTmp, RhClauses,
                  ExchangePropertiesVarListOut, RhOpSet),
      borrowAllObjects(ExchangePropertiesVarIn),  ;Getting the objects for the Ex-Co
      applyExchangeClause(LhOpSet, ExchangePropertiesVarIn,   ;Test of Premise
                  ExchangePropertiesVarTmp),!,               ;Cut Symbol if true
      applyExchangeClause(RhOpSet, ExchangePropertiesVarTmp,ExchangePropertiesVarOut),
      returnAllObjects(ExchangePropertiesVarOut). ;Giving the objects back to CompSt

getExchangePropertiesVarList([(PropertyRef,View, ExName)|RestList],
                  [(PropertyRef, View, ExName, _Variable)   |RestNewList]).
getExchangePropertiesVarList([],[]).

translateExchangeClause(ClassObjectList, ListIn,[Clause|RestClauses], ListOut,
                  [(ListIn,OP,ListNew)|RestOPs]) :-
      Clause =.. [(Class.Method), ExNameList, ParameterList],
      getClassObject(ClassObjectList, Class, Object),
      getExNameVariables(ListIn, ExNameList, ExVarList),
      methodIsAProperOperation(Class, Method), !,     ; method is a proper operation
      OP = op(Object, Method, ParameterList, ExVarList, ExVarListNew),
      retainExNameVariable(ExVarListNew, ExNameList, ListNew),
      translateExchangeClause(ListNew, RestClauses, RestOps).

translateExchangeClause(ClassObjectList , ListIn,[Clause|RestClauses], ListOut,
                  [(ListIn,TEST,ListIn)|RestOPs]) :-
      Clause =.. [(Class.Method), ExNameList, ParameterList],
      getClassObject(ClassObjectList, Class, Object),
      getExNameVariables(ListIn, ExNameList, ExVarList),
      TEST = test(Object, Method, ParameterList, ExVarList),
      translateExchangeClause(ListIn, RestClauses, RestOps).

translateExchangeClause(ListOut,[], ListOut, []).

borrowAllObjects([(PropertyRef, View, ExName, Object)|RestList]) :-
      mergeProperty(PropertyRef),!,
      getMergePropertyObject(PropertyRef, MergeObject, ClientObject, ServerObject),
      op(MergeObject,Export,[View], [MergeObject, ClientObject, ServerObject],
                  [Object]),
      borrowAllObjects(RestList).

borrowAllObjects([(PropertyRef, View, ExName, Object)|RestList]) :-
      getCharPropertyObject(PropertyRef, Object),
      returnAllObjects(RestList).

borrowAllObjects([]).


returnAllObjects([(PropertyRef, View, ExName, Object)|RestList]) :-
      mergeProperty(PropertyRef),!,
      op(MergeObject,Import,[View], [MergeObject, ClientObject, ServerObject, Object],
                  [MergeObject, ClientObject, ServerObject]),
      setMergePropertyObject(PropertyRef, MergeObject, ClientObject, ServerObject),
                  returnAllObjects(RestList).

returnAllObjects([(PropertyRef, View, ExName, Object)|RestList]) :-
      setCharPropertyObject(PropertyRef, Object),
      returnAllObjects(RestList).

returnAllObjects([]).




applyExchangeConstraint([(ListIn,OP_TEST,ListTmp)|RestLhOpSet], ListIn, ListOut) :-
      call(OP_TEST),
      applyExchangeConstraint(RestLhOpSet,ListTmp,ListOut).

applyExchangeConstraint([], ListOut, ListOut).
```

Code 6 - Application and Translation of Exchange Constraints

Example 19 **Telecommunication** with Gateway **(continuation2)**

Code 7 - CompSt for telecommunication with Gateway, shows the resulting Composite Service data structure (CompSt, cf. Definition 24) of Example 17, which was developed in Example 18, cf. Fig. 38. The object dictionary containing 20 entries is listed. For simplification, it is indexed with the object references given in the Property Descriptions of a Selected Mode (ModeProperty) and of a merged Service Port (MeProperty). Entries for Exchange Property Objects, used during the application of Exchange Constraints, are not listed, because the Exchange Constraints with a fulfilled premise were all successfully applied ("fired") at a positive end of a Composition Process. The constraints were anyway part of the extended Composite Structure (CompSt-plus, cf. Definition 25). Thus, only two kinds of Property Objects are listed in the object dictionary: Char Property Objects and Merge Property Objects. The latter are referenced in the Property Descriptions (MeProperty) of a merged Service Port (Merged-SP). They are originated from the Merge Constraints and discussed in the section 13.5. Every Property Object is supposed to handle the final implementation instance of its Service Property after the successful Composition Process. These implementation instances will hold the information of the Service Properties, which complies with all constraints applied in the Composition Process. The information is helpful for the subsequent phases of the life cycle of the discovered Composite Service. At least, it can be said that this Composite Service is a Service Candidate fulfilling all criteria examined by ACTAS.

The Definition 27 defined the Composition Graph of a Composite Service in ACTAS. The adjacency lists of this graph can be found in the entries of the selected Service Modes (Selected-SM) of the Composite Service. Selected-SM$_1$ and Selected-SM$_2$ come from the initialising ASOs discussed in Example 17. The three other ones are the selected Service Modes of the Component Services shown in Fig. 38. The entries of the selected Service Modes and the merged Service Ports contain mainly references. A selected Request Mode of an ASO contains a reference to the original Service Request (SRe$^{\text{Ref}}$) and the Request Mode of the ASO built from the Client Request of the Service Request (ASO-RM$^{\text{Ref}}$). The selected Service Mode of a Service Offer of a Component Service references the SOER, it was built on. The merged Service Ports have references to the as compatible recognized Service Ports of the selected Service Modes of the (A)SOs (cf. section 13.3 - Step2: Initialisation of the Composite Structure). The field "Res-Info" was discussed in the possible post processing of the Composition Process (cf. section 13.7). It can be used for the reservation of the service by the FA.

$$\text{CompSt}_{\text{DFC}} = \left(\text{CompSt}^{\text{ID}}_{Gat},\ \text{Selected-SM}^{Set},\ \text{ComProperty}^{Set},\ \text{Merged-SP}^{\text{Set}},\text{Object}^{Set}\right)$$

$$\text{Selected-SM}_1 = \left(\text{RM}_1^{\text{ID}},\ \text{SRe}_1^{\text{Ref}},\ \text{ASO-RM}_1^{\text{Ref}},\ \text{ModeProperty}_1^{\text{Set}}, \{\text{Mer-SP}_1\}\right)$$
$$\text{Selected-SM}_2 = \left(\text{RM}_2^{\text{ID}},\ \text{SRe}_2^{\text{Ref}},\ \text{ASO-RM}_2^{\text{Ref}},\ \text{ModeProperty}_2^{\text{Set}}, \{\text{Mer-SP}_2\}\right)$$
$$\text{Selected-SM}_3 = \left(\text{SM}_3^{\text{ID}},\ \text{SOER}_3^{\text{Ref}}, \text{SO-SM}_3^{\text{Ref}},\ \text{Res-Info}_3, \text{ModeProperty}_3^{\text{Set}}, \{\text{Mer-SP}_1, \text{Mer-SP}_3\}\right)$$
$$\text{Selected-SM}_4 = \left(\text{SM}_4^{\text{ID}},\ \text{SOER}_4^{\text{Ref}}, \text{SO-SM}_4^{\text{Ref}},\ \text{Res-Info}_4, \text{ModeProperty}_4^{\text{Set}}, \{\text{Mer-SP}_2, \text{Mer-SP}_4\}\right)$$
$$\text{Selected-SM}_5 = \left(\text{SM}_5^{\text{ID}},\ \text{SOER}_5^{\text{Ref}}, \text{SO-SM}_5^{\text{Ref}},\ \text{Res-Info}_5, \text{ModeProperty}_5^{\text{Set}}, \{\text{Mer-SP}_3, \text{Mer-SP}_4\}\right)$$
$$\text{Merged-SP}_1 = \left(\text{Mer-SP}_1^{\text{ID}},\ \text{Cl-SP}_1^{\text{Ref}},\ \text{Se-SP}_1^{\text{Ref}},\ \text{true},\ \text{MeProperty}_1^{\text{Set}}\right)$$
$$\text{Merged-SP}_2 = \left(\text{Mer-SP}_2^{\text{ID}},\ \text{Cl-SP}_2^{\text{Ref}},\ \text{Se-SP}_2^{\text{Ref}},\ \text{true},\ \text{MeProperty}_2^{\text{Set}}\right)$$
$$\text{Merged-SP}_3 = \left(\text{Mer-SP}_3^{\text{ID}},\ \text{Cl-SP}_3^{\text{Ref}},\ \text{Se-SP}_3^{\text{Ref}},\ \text{false},\ \text{MeProperty}_3^{\text{Set}}\right)$$
$$\text{Merged-SP}_4 = \left(\text{Mer-SP}_4^{\text{ID}},\ \text{Cl-SP}_4^{\text{Ref}},\ \text{Se-SP}_4^{\text{Ref}},\ \text{false},\ \text{MeProperty}_4^{\text{Set}}\right)$$

$\text{Object}_1 = (\text{SeqD\_Object}_1, \text{Char\_Property\_Object}_1)$
$\text{Object}_2 = (\text{SeqD\_Object}_2, \text{Char\_Property\_Object}_2)$
$\text{Object}_3 = (\text{Cl\_De\_Object}_1, \text{Char\_Property\_Object}_3)$
$\text{Object}_4 = (\text{Se\_De\_Object}_1, \text{Char\_Property\_Object}_4)$
$\text{Object}_5 = (\text{Me\_De\_Object}_1, \text{Merge\_Property\_Object}_1)$
$\text{Object}_6 = (\text{Cl\_AQu\_Object}_1, \text{Char\_Property\_Object}_5)$
$\text{Object}_7 = (\text{Se\_AQu\_Object}_1, \text{Char\_Property\_Object}_6)$
$\text{Object}_8 = (\text{Me\_AQu\_Object}_1, \text{Merge\_Property\_Object}_2)$
$\text{Object}_9 = (\text{Cl\_De\_Object}_2, \text{Char\_Property\_Object}_7)$
$\text{Object}_{10} = (\text{Se\_De\_Object}_2, \text{Char\_Property\_Object}_8)$
$\text{Object}_{11} = (\text{Me\_De\_Object}_2, \text{Merge\_Property\_Object}_3)$
$\text{Object}_{12} = (\text{Cl\_AQu\_Object}_2, \text{Char\_Property\_Object}_9)$
$\text{Object}_{13} = (\text{Se\_AQu\_Object}_2, \text{Char\_Property\_Object}_{10})$
$\text{Object}_{14} = (\text{Me\_AQu\_Object}_2, \text{Merge\_Property\_Object}_4)$
$\text{Object}_{15} = (\text{Cl\_Spe\_Object}_1, \text{Char\_Property\_Object}_{11})$
$\text{Object}_{16} = (\text{Se\_Spe\_Object}_1, \text{Char\_Property\_Object}_{12})$
$\text{Object}_{17} = (\text{Me\_Spe\_Object}_1, \text{Merge\_Property\_Object}_5)$
$\text{Object}_{18} = (\text{Cl\_Qu\_Object}_1, \text{Char\_Property\_Object}_{13})$
$\text{Object}_{19} = (\text{Se\_Qu\_Object}_1, \text{Char\_Property\_Object}_{14})$
$\text{Object}_{20} = (\text{Me\_Qu\_Object}_1, \text{Merge\_Property\_Object}_6)$

$\text{ModeProperty}_{3,1} = (\text{Feature, SequenceDiagram, SeqD\_Object}_1)$
$\text{ModeProperty}_{5,1} = (\text{Feature, SequenceDiagram, SeqD\_Object}_2)$

$\text{MeProperty}_{1,1} = (\text{Loc-Auth, y, Me\_De\_Object}_1, \text{Cl\_De\_Object}_1, \text{Se\_De\_Object}_1)$
$$\text{MeProperty}_{1,2} = \left(\begin{array}{c} \text{Audio-Com, Audio-Quality, Me\_AQu\_Object}_1, \text{Cl\_AQu\_Object}_1, \\ \text{Se\_AQu\_Object}_1 \end{array} \right)$$
$\text{MeProperty}_{2,1} = (\text{Feature, Dependency, Me\_De\_Object}_2, \text{Cl\_De\_Object}_2, \text{Se\_De\_Object}_2)$
$$\text{MeProperty}_{2,2} = \left(\begin{array}{c} \text{Audio-Com, Audio-Quality, Me\_AQu\_Object}_2, \text{Cl\_AQu\_Object}_2, \\ \text{Se\_AQu\_Object}_2 \end{array} \right)$$
$\text{MeProperty}_{3,1} = (\text{Phone, Speed, Me\_Spe\_Object}_1, \text{Cl\_Spe\_Object}_1, \text{Se\_Spe\_Object}_1)$
$\text{MeProperty}_{3,2} = (\text{Phone, Quality, Me\_Qu\_Object}_1, \text{Cl\_Qu\_Object}_1, \text{Se\_Qu\_Object}_1)$

Code 7 - CompSt for telecommunication with Gateway

EVALUATION

14 ACTAS

ACTAS is a framework for Service Discovery and Service Composition, which categorizes services, their ontologies, and their constraint checking algorithms on a meta-level. The evaluation discusses the hypothesis, especially how ACTAS can ease the composition of services.

The evaluation begins with some common statements in the context of Service Discovery and Service Composition. A starting question could be: **Is ACTAS just another Service Discovery environment like WSMO, OWL-S for Semantic Web Services or simply like UDDI for Web Services?**

The State-of-the-Art chapter showed that existing Service Discovery environments select services mainly through their functional description. Only in a second step, non-functional descriptions, extended concepts, and features are involved. Generally, they make a firm distinction between the description of elements and executable technologies (of the Service Discovery environment as well as the services). In ACTAS, the Semantic Characteristics principally allow Service Requests for any aspect of services, a functional description is not obligatory. A Service Request can be freely composed of several Semantic Characteristics and a Semantic Characteristic can involve several aspects of services. Additionally, an agreement on common matching and mediation algorithm is guaranteed, since the properties of the Semantic Characteristics are associated with appropriate resources, i.e. ontologies and algorithms. Thus, ACTAS is a framework, which can connect services on the basis of any commonly agreed aspects linked with fitting resources. Separate repositories of services like in Meteor-S are not any longer necessary. Nevertheless, the separation between the Service Description and the execution environment of the actual services is preserved as demanded by Berners Lee (cf. Definition 3). This means that ACTAS does not change the functional behaviour of the services.

This leads to the question: **What is a service in ACTAS?** A service in ACTAS is not necessarily a representation of exactly one service (electronic, manual, or physical, cf. State-of-the-Art). In the case of directed Service Composition, a Service Description in ACTAS can have several IN ports and OUT ports. Every Server Port, i.e. Service Port with the "direction" Option-Slot having the attribute "IN", represents principally a service. ACTAS does not know whether this is an Abstract or a Concrete Service. This information is not relevant for ACTAS. In the case of non-directed Service Composition, a service in ACTAS can become identical with the service offering facility. For instance in Case Study 2: Distribute Feature Composition (DFC), the selected Service Offer Modes of the Composite Structure (cf. Fig. 51) represent a telephone exchange, closer tested and addressed through the General Characteristic "Feature".

Summarizing, it can be said that each Service Mode of a Service Description in ACTAS relates interface descriptions, which can stand for offered and requested services. However, ACTAS does not describe process-like relationships between the interfaces. The Composition Process will select a Service Mode, i.e. a certain constellation of interfaces.

How does ACTAS take advantage of existing algorithms? ACTAS works on two levels of abstractions of a service: (1) the service classified through Semantic Characteristics, and (2) the Service Properties wrapped in the semantic context of these characteristics as Char Properties. ACTAS is an open system, i.e. the Semantic Characteristics are distinct through their (URL) references to ontological repositories. This is important, in order to have a reliable application of the principal compatibility between services (cf. Definition 11). The declarations of Char Properties of the Semantic Characteristics can be adapted, without changing the Service Descriptions of ACTAS. The Service Designer does not have to use every Char Property. It should be possible to check whether the Char Properties are initialised, i.e. used. Only used Service Properties are kept in the data structures of the C-Model. Exchange Constraints defined inside of the Semantic Characteristics could verify if a current setting of the variables is acceptable in the semantic context of the characteristic.

As shown in Definition 4, the Char Properties get associated with algorithms handling their information. The kept information of a Char Property depends on the semantic context of its characteristic. It can be the temperature of a room; the GPS coordinates of a location, or the functional description of a service through WSML. For all different kinds of information, algorithms for their management exist. The Semantic Characteristic can declare additional constraints or ontologies valid in its semantic context through Value Constraints. The algorithms offer methods for taking on-board these constraints and ontologies for their internal constraints as discussed in section 13.1. In this way, ACTAS uses given algorithms and information adapted to a semantic context.

The Char Properties of Compatibility Characteristics (cf. Definition 4) are additionally associated with the Merge Property Classes. The rule of the principal compatibility ensures that a Char Property becomes only "merged" with a Comparable Property (cf. Definition 12), i.e. a Char Property of the same name declared in the same Semantic Characteristic. In the case of a Capability Description of SWS kept in a Char Property, the algorithms like [KluFri et al.2009; KluKau2009] for matching of WSMO and OWL-S services could be used. The merge algorithm of ACTAS can also include mediation. The information for the mediation could be kept in the Char Property. The applied matching and mediation should become content of the Semantic Description of the wrapping Compatibility Characteristic. The openness and adaptability of ACTAS is shown again through the fact that the Service Property information could be extended and its associated algorithms improved without implying a change of the Service Descriptions of ACTAS. This fact is valid as long as the once published methods are still supported.

Only the Exchange Constraints enumerate in their term "ExchangeClasses" Exchange Property Classes directly (cf. Definition 18). Nevertheless, these are only references to the proposed ontological repository of Property Classes (cf. section 9.2). Therefore, even these methods can be transparently improved. Through the controlled publication, the methods become commonly agreed and trustful algorithms useful in Service Discovery and Service Composition. On the level of abstraction of Service Properties, this is a necessary step for the future realisation of the Autonomic SOC.

How does the B2C and B2B concept in e-business compare to the supported Service Composition in ACTAS? ACTAS introduced a special kind of Compatibility Characteristics, the Request Characteristics, in order to describe the compatibility interfaces towards Service Requests. In the thesis, this was several times stated as a B2C like connection. As the Case Study 3: Supply Chain, B2B Integration shows, it cannot be spoken about congruence in the use of these terms in ACTAS and e-business. To put it simple: An e-business service of the B2B integration could be requested through a Service Request in ACTAS addressed through a distinct Request Characteristic. Nevertheless, the purpose of the distinction between Request Characteristic and normal Compatibility Characteristic used in the Service Ports follows similar ideas. The examples of Technical Services can be seen as a further proof of the feasibility of these ideas. The user interface described through the Request Characteristics shows only the services interesting for a certain user group. For instance audio communication, which is covered with the Request Characteristic “Audio-Com” in the case study, is just addressing potential Service Clients of a telecommunication service. The Service Properties describing the technical side of for instance telephone exchanges are wrapped with a Compatibility Characteristic “Phone” describing the non-directed connection between two exchanges. It is not declared as a Request Characteristic. Thus, the latter connection for the realisation of the audio communication is transparent for the Service Clients, because it is realised through B2B like Service Composition in the Composition Process. The Case Study 1: Technical Services with translation has a closer look at these services.

How does ACTAS adapt to the policies of the involved parties and supports them? Firstly, ACTAS is conceptually based on a MAS introduced in the System Environment (chapter 8). The roles of the software agents are congruent with the roles of the involved parties: Service Provider – Facility Agent, Service Requester – Request Agent, and Service Trader – Trader Agent. The introduced Service Client is ideally represented through a Personal Agent. The pro-active behaviour of the agents is determined through the correlation of roles. The Composition Agent is created by the Request Agent. Therefore, its behaviour can be determined by the user / application side. Alternative algorithms of the Composition Process are discussed later in this chapter. The Composition Process itself can be influenced by the Service Designer through the Option-Slots in the diverse environments (Env) of the Service Descriptions and Service Requests. The influence of the “Translation” Option-Slot is covered in this chapter. The Facility Agent can use the distinction between Service Templates and SOERs for its resource management. It can adapt through the SOERs the Service Templates to the current available resources, i.e. only valid Service Modes, which can be realized, are enumerated in the SOER. The

SOER can contain additional Value Constraints for the Service Properties. The SOER referenced in a selected Service Mode in the resulting Composite Structure (cf. Definition 24) can be used for the SLA between Service Requester and Service Provider. This would mean that the Facility Agent would confirm the reservation of the service and its resources on the basis of this SOER. The CompSt contains an additional field "Res-Info" for keeping information in this direction. It could be used in the later phases of the life cycle (Service Grounding and Deployment), in order to be sure that the service and resources were reserved so that it can be realised. The clear correlation between service, Service Provider, and the Facility Agent enables these considerations.

What is with the support of the non-functional property "availability" in ACTAS? Considerations for the availability control by the Facility Agents were just stated in the previous paragraph. Availability is a non-functional property, which should be directly supported by a

Topic	Point of interest
System Environment	• What is the ideal environment? • What art the limitations?
Service Design and Service Description	• Characteristics allow a multi-dimensional semantic description • Simple Description of Compatibility with characteristics • Specific Characteristics for User Requests • Characteristics can be used as "building blocks" • Exchange Constraints, define correlations between Service Properties • Directed and non-directed Service Compatibility Descriptions
Service Request	• The combination of several Client Request • The possibility to define constraints for the whole Service Request and Composite Structure in its Common Part
Service Discovery • Service Trading • Service Matching • Service Ranking • Service Mediation	• No domain specific repositories necessary • Inclusion of other Trading Environments • Composition Agent performs an application specific algorithm • Selection and composition of Service Modes through the application of the rule of principal compatibility, i.e. compatible Service Modes have the same set of characteristics. • Application of constraints through algorithms linked with the Service Properties • Merge Constraints allow the inclusion of matching algorithms and mediating • Translation and mediation in ACTAS with Exchange Constraints

Table 21 - Points of Evaluation

framework like ACTAS, in order to ensure that after the performance of the later phases of the life cycle the service is still available. Nevertheless, the availability is certainly a criterion of the second aspect of services and should be considered by ACTAS Administrators in the definition of Semantic Characteristics. In the appendix, several WSML ontologies for the description of non-functional properties are listed. The nfp-ontology for availability is shown in section B-1 of the appendix. The ontology defines several properties for the concept "Availability": "isAvailableAt", "isAvailableDuring", and "isAvailableTo". This concept is extended to the concept "RequestAvailability" providing the additional properties: "forRequest", "hasNegotiableTime", and "isContinuouslyAvailable". The ACTAS Administrator could directly use these concepts and their properties for the introduction of Semantic Characteristics. Compatibility/Request Characteristics would ensure the availability of a service. A Service Request, which had only this Request Characteristic in its Request Port, would just look for all services, which have this Semantic Characteristic in their Service Port and fulfil the Merge Constraints. In other words, it would discover all available services according to the given criteria. The function of the services does not matter in this Service Request. This flexibility of compatibility in ACTAS was already previously stated in this section. In this chapter, the implementation of the algorithms associated with the Char Properties through generic frameworks for reasoning with WSML is discussed.

In Table 21 - Points of Evaluation, further statements about ACTAS, sorted by phases of the life cycle, are made. These are discussed in the following sections. Afterwards four case studies are done.

14.1 Environment of ACTAS

The ideal environment of ACTAS was already discussed in the description of the System Environment (chapter 8). The different parties involved in the Service Oriented Computing are represented through an autonomic Software Agent. In an optimal case, the Facility Agent publishes ST and SOER. On the one hand, the FA uses the SOER, in order to adapt the ST to the currently available resources. The Facility Agent is doing the resource management and the reservation of resource for the requesting application. On the other hand, the Service Trader can perform a Service Trading with the best possible Service Modes of a service. The selected Service Mode is the first information for the following phases of the life cycle of the discovered services done by the systems, which are offered by the Service Providers of the selected Component Services. ACTAS delivers further information through the Service Properties, which can have their external implementation instances. Web Services are likely for the implementation of these external implementation instances.

In Case Study 2 (cf. chapter 16), an implementation of the support of Domain-Specific Languages (DSL) based on SOA, published in [ShiAda et al.2010], was discussed, because it illustrates in which way the algorithm associated with the Service Properties could be realised. In the declarative environment based on SICStus Prolog, the implementation instances are accessed through Property Objects. "The SICStus Objects package enables programmers to write object-oriented programs in SICStus Prolog. The objects in SICStus Objects are modifiable data structures that provide a clean and efficient alternative to storing data in the Prolog database." [Car2009 p. 383]

In the C-Model, the access of implementation instances through Property Objects was covered. It was explained that the clowning of the Property Objects is necessary, in order to support the backtracking algorithm of the declarative environment. The clowning of the Property Object implied a new implementation instance, which had also to be "clowned". In Fig. 39 - The Object Dictionary, the treatment of the stacked information is schematically shown. The object dictionary (Object^{Set} in CompSt, cf. Definition 24) holds the Property Objects indexed by the references of its Service Properties or Merge Properties, respectively (cf. Definition 15). The object dictionary also contains the Exchange Property Objects, for the access of the implementation instances of the methods used in the Exchange Constraints. In the popular book of Prolog programming, "The Art of Prolog" by Sterling and Shapiro [SteSha1994], the object dictionary is a typical incomplete data structure, because the Property Objects are permanently replaced with new ones. Thus, the difference-lists (described in chapter 15 of [SteSha1994]) are used.

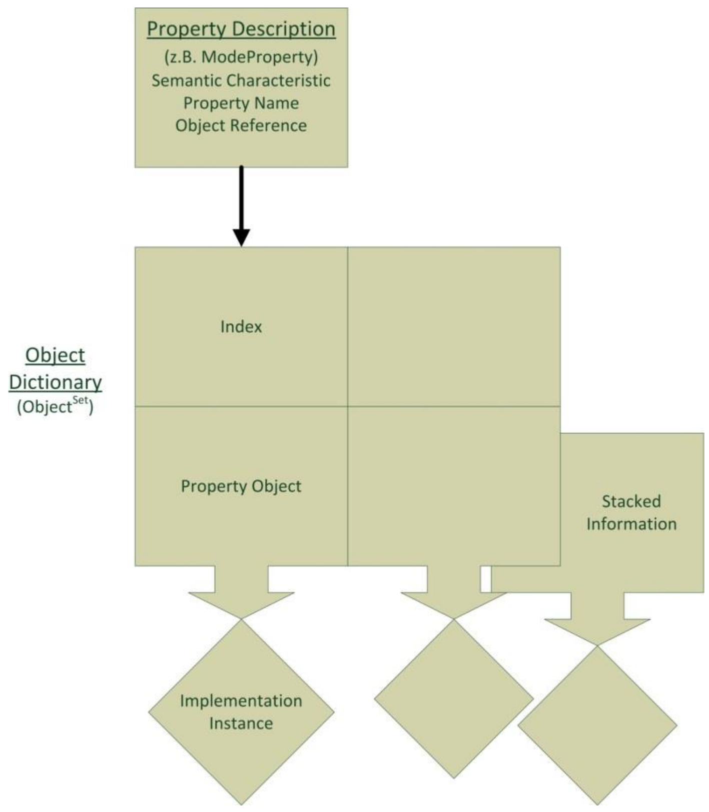


Fig. 39 - The Object Dictionary

In the thesis, it was decided not to describe in closer detail, how this wrapping and clowning of the access of the implementation instances can be done. This is future research. The creation and management of the handles of external implementation instances, realised through the Prolog objects, goes beyond the scope of this dissertation. An alternative approach to Web Services would be the supply of a Prolog module for the implementation of the methods of the Property Class. This module could be saved in the ontological repository of the Property Classes. Such a Prolog module can directly be integrated into the object concept of Prolog, in order to have separated data for each object instance. The module system of SICStus Prolog is quite elaborated and was part of my research for my diploma [BeKlMe2000]. In this way, the access and implementation of the Property Classes could become transparent for ACTAS, secure and highly adaptive. However, it can be expected that the solutions using extensively external implementation instances based on SOA, will not be very performing. The authors of the previously mentioned publication about a FDL realising SOA approach [ShiAda et al.2010] spoke about their experiences in this direction.

In the appendix, nfp-ontologies, written in WSML, are listed. Possible Semantic Characteristics for non-functional properties based on these ontologies are covered in the subsequent section. In this section, another realisation of the implementation instances shall be sketched.

The WSML2Reasoner[6] framework is a generic, flexible architecture for reasoning with the different variants of the WSML family. It was used for the realisation of SESA introduced in [FeKeZa2008]. The fact that WSML is based on (theoretically and practically) well-studied knowledge representation paradigms, for which various systems have already been implemented and tested, allows the framework design of WSML2Reasoner itself and the integration of external reasoning components. Thus, this reasoner could be addressed from the declarative environment of ACTAS.

Let us assume that an ACTAS Administrator created the Compatibility Characteristic "Availability" as it appears in the nfp-ontology in section B-1 in the appendix. He could have also written the declaration of the three mentioned properties ("isAvailableAt", "isAvailableDuring", and "isAvailableTo") as three separate Char Properties. The ontology uses other ontologies for the type definition of the properties and can include some additional constraints for the properties of a concept. The ontology will always be updated, when a change to the declarations becomes necessary. Therefore, it would be a good idea to use the ontology itself, in order to implement and check the constraints of Char Properties of Semantic Characteristics, which are based on a concept of a WSML-ontology. Thus the Char Property Classes could be implemented through the WSML2Reasoner, in order to deal with the Char Properties and their internal constraints according the ontology (as shown in rule (intOp)). The current ontology has to be simply set by a Value Constraint using an initialisation method of the Property Class. In this way, algorithms could be implemented, which could work as Property Classes for several Char Properties, which were declared on the base of WSML ontologies. Thus, the number of implemented algorithms for the Char, Merge, and Exchange Property Classes can be extremely reduced.

14.2 Service Design - "Building Blocks" of ACTAS

The experience with UDDI as a central repository showed that strict decoupling of resource development like semantic descriptions has to be observed by a Service Discovery environment. The resulting heterogeneity issues between resources have to be resolved through a centrality of mediation (cf. [FeKeZa2008 section 44]). This is for instance a central vision of WSMO. For this purpose WSMO inherits the concept of Universal Resource Identifier (URI) from the Web as the mechanism for unique identification of resources including the mediation algorithms. The

[6] tools.deri.org/wsml2reasoner/index.html

centralized mediation does not only address ontologies, which are the core of the WSMO meta-model, but also includes the mediation between service and goal classes.

Independently of WSMO, ACTAS developed a similar essential vision. The Service Discovery environment cannot rely on the SOA alone, i.e. the publication, trading, and discovering of services based on informal, functional, possibly semantically enhanced Service Descriptions. It needs a centralized semantic categorisation of services and the common agreement on algorithms applicable in the resulting semantic contexts of the categories. The centralisation of Semantic Characteristics and Property Classes are achieved through the ontological repositories of these entities. Like WSMO, ACTAS is using references, e.g. URI, in order to have a unique addressing of the entities, which is important for the principal compatibility and the application of the right algorithms.

OWL-S allows the use of descriptions in other languages (e.g. constraints written in KIF), but it does not integrate algorithms like. The semantic hierarchy of Web Services is allowed, but I do not know any application. ACTAS shows that the classification of services cannot be only hierarchically, but has to include several aspects. WSMO restricts its centralized algorithms to the mediation. It is worthwhile to have a closer look at the mediation of WSMO, in order to reach a better delimitation towards ACTAS.

Three levels of mediations are supported by WSMO:

1. Data-level mediation - mediation between heterogeneous data sources; mostly ontology integration
2. Protocol-level mediation - mediation between heterogeneous communication protocols; i.e. the choreography entry in WSMO is concerned
3. Process-level mediation – mediation between heterogeneous business processes, i.e. the orchestration entry in WSMO is concerned

WSMO defines mediator types between its components: OO Mediators, GG Mediators, WG Mediators, and WW Mediators (O – Ontology, G – Goal, W – (Web) Service). The OO Mediators resolve mismatches between ontologies and provide mediated domain knowledge specifications to the target component. A GG Mediator connects goals and allows the creation of a new goal from existing goals. The WG Mediators links a Web Service to a goal, resolving terminological mismatches, and stating the functional difference (if any) between both. A WW Mediator is used to establish interoperability between Web Services that are not interoperable a priori.

ACTAS extends these concepts of WSMO, with adding reliable semantic contexts for the centralized algorithms. These semantic contexts are called Semantic Characteristics and they gain their semantic with relations to ontologies, which categorise various aspects of services (cf. State-of-the-Art, section 2.3 and 9.1). Thus, ACTAS is working on two levels of abstractions of services. The categorisation of services through Semantic Characteristics and their descriptions through Service Properties wrapped in the semantic context of the characteristics. In Table 27, assumed ontologies for the declaration of concepts based on the four aspects are introduced, e.g.:

- An ontology for application domains ($\mathsf{Ontology}_{\mathsf{Domain}}$),
- an ontology for the Service Design ($\mathsf{Ontology}_{\mathsf{Design}}$),
- and an ontology for non-functional concepts ($\mathsf{Ontology}_{\mathsf{nfp}}$).

Examples for existing nfp-ontologies are given in appendix chapter B. In the appendix chapter A, tables enumerate declarations and descriptions of Semantic Characteristics used in the thesis. The Semantic Characteristics and the Property Classes are referenced through simple names. Compatibility Characteristics and General Characteristics become distinguishable through the "is-a" relation in their semantic description. Request Characteristics are Compatibility Characteristics, which can have the additional relation "Can_be_used_by". This relation can be specified in the semantic description of the RCh. It relates the services classified by the Request Characteristics with user groups defined in the assumed ontology $\mathsf{Ontology}_{\mathsf{User}}$. (The RCh concept in the "is-a" relation is a sub-concept of CCh.)

In the following paragraphs, considerations for the introduction of Semantic Characteristics and the Service Design are discussed. The Semantic Characteristics used as "building blocks" are discussed in the first and the second case study. In this context, the "Works-with" relation is mentioned again. It relates Semantic Characteristic working as "building blocks" together. Such Semantic Characteristics are linked through Exchange Constraints. The "Works-with" relation has three fields. The first two fields state the Semantic Characteristics. The third field contains the identification of the linking Exchange Constraint of the secondly stated Semantic Characteristic. The comparison with the mediation of WSMO is continued with the discussion of Translation Offers (Service Offers, which use Exchange Constraints for the mediation between the Semantic Properties of Semantic Characteristics) in Case Study 2: Distribute Feature Composition (DFC).

It is up to the ACTAS Administrator to decide, which ontologies he uses for the semantic description of characteristics. Furthermore, he has to decide about the publication of Semantic Characteristics and Property Classes. Ontologies like the nfp ones in appendix B can help for the creation. Some concepts could be introduced as Semantic Characteristics; others could be better implemented as Property Classes. The Price Ontology is a candidate for a Property Class, since it is mainly used as a type of attributes in other ontologies; whereas the Provider Ontology describes an annotating non-functional parameter, which is of interest for the Service Selection. Therefore, the administrator decides that it is worthwhile to introduce a General Characteristic "Provider-Description". The semantic descriptions can be found in Table 30 to Table 34. In the subsequent sections Semantic Characteristics in the context of the aspects of services are discussed.

14.2.1 Simple Semantic Characteristics

The location of a service is often decisive about its usability, e.g. for the selection of a communication facility service or the use of the public transport. Therefore, the ACTAS Administrator will consider the introduction of a Compatibility Characteristic, which allows the testing of the principal compatibility in the context of service location. The principal compatibility in this case means that the service declares that it provides the location information in a commonly agreed way, and that it allows a pre-selective access of this information as well as a matching and mediation of location descriptions. For this purpose, the semantic description of the CCh "Location" in equation (22-17) of Table 34 - Request Characteristics (RCh) is defined through the nfp-ontology concept "GeographicalRegion" (cf. appendix section B-3 - NFP-Ontology for location (locative)).

For the Service Discovery, it is essential to have information about the kind of the functional Service Description supported by a Service Offer (4th aspect of services – Service Design (phase 1)) or used by the Service Request. The Service Design is concerned with the Service Description. Therefore, Semantic Characteristics, wrapping properties with WSMO and OWL-S Service Descriptions, will be related in their semantic description with concepts from ontologies in this direction (the assumed $\text{Ontology}_{\text{Design}}$ in this case). The Table 34 contains the Request Characteristics "WSMO", "OWL-S_with_IOPE", and "OWL-S-Geodata" in the equations (22-21), (22-18), and (22-19). The first one wraps a general WSMO component description. It classifies all services using the SWS Service Description formalism WSMO; whereas the Compatibility Characteristics "OWL-S_with_IOPE" and "OWL-S-Geodata" address only the services with a SWS Service Description in OWL-S. The former is declared in its semantic description as a characteristic for a general OWL-S using IOPE. The latter connects the imagined domain Geodata with the OWL-S Service Description. These associations are shown with concepts of a design-ontology, which are named as "general".

The meaning of "general" is up to this design-ontology classifying the different (functional) Service Description formalism like WSMO, OWL-S, WSDL or UML. The design-ontology might introduce more specific concepts, in order to describe for instance that a Service Description contains simply capability descriptions, i.e. only parts of the Service Description formalism. For one kind of Service Description, several Compatibility Characteristics can be introduced due to alternative matching algorithms and domain specific application ontologies. Accordingly, the logical expression of the semantic description will include several concepts of ontologies covering multi dimensions of the semantic, i.e. all aspects of services (cf. section 2.3). The addressing of specific matching algorithm could be reflected in the semantic description through the relation with concepts of an ontology classifying the matching ($\text{Ontology}_{matching}$ in Table 27 - Assumed Ontologies for the classification, phase 3 of the life cycle, i.e. the 4th aspect of services).

The listed Compatibility Characteristics for the Service Description could wrap no Char Properties. In this case, they simply would be "carrier" of their semantic description, in order to select services supporting their description formalism. The Service Discovery would completely take place in external environments. On the other hand and this is the more likely case, they have Char Properties and the Service Designer can decide whether he wants to use them as Service Properties. E.g. the RCh "WSMO" could declare a Char Property holding the information of a WSMO component. This component can be initialised through a Value Constraint, giving the URL of the WSMO component. In a Service Request, the component would be a goal, in the other case a service component. The algorithms of the Property Class associated with the Char Property should be able to execute the mediators as introduced above, since every component in WSMO can be declared with the access on certain mediators. The advantage of ACTAS becomes apparent as the Service Designer can rely on the associated algorithms. They will be capable to deal properly with the complicated handling of WSMO components. Furthermore, he knows that the merge, which includes mediation and matching, of comparable WSMO components (most likely goal and service) are done in the right way. The Exchange Property Classes provide additionally a bunch of algorithms for the mediation / translation of the information of the SWS components in other formats. All these is done through centralized, referenced algorithms, which offer a trustful and commonly agreed behaviour, since they are published in the managed ontological repositories of Property Classes.

The Table 34 - Request Characteristics (RCh) also lists an RCh "Domain-Geodata" (equation (22-14)). The assumed domain "Geodata" could be occupied with the processing of geographic data. The association of the Service Description with a certain domain is shown through this Semantic Characteristic (1st aspect of services). A service associated with the Compatibility Characteristic "Domain-Geodata" is just asking for Service Offers in this domain. In the context of some applications, the association with just this Compatibility Characteristic might be enough as a pre-selecting criterion, in order to fill domain-specific repositories with up to date Service Offers or simply Service Providers. Alternatively, a service can be associated with several Compatibility Characteristics. For instance, a service could hold the Compatibility Characteristics "OWL-S_with_IOPE" and "Domain-Geodata". The service instances categorised through this combination of characteristics are the intersection of the service instance sets categorised through a single characteristic. In the given example, only services in the Geodata domain using OWL-S Service Descriptions are addressed through the combination.

Obviously, a service interface can be described in several ways. This leads to the discussion, which option is preferable. On the one hand, the semantic description of a Semantic Characteristic allows more constructors for the combination of ontological concepts. On the other hand, such a Semantic Characteristic might be too restrictive and a combination of characteristics in the Service Description could be more adaptive. However, when the discussion is extended to the wrapped Char Properties, then a Semantic Characteristic with a more complex semantic description can also include more sophisticated properties with more specialised algorithms. In the context of Alternative Service Requests (cf. section 14.3), the possibility of the description of services/requests in ACTAS in different ways is sketched anew.

The figures Fig. 38 and Fig. 28 include the nfp-criterion "reliability". The semantic description of an RCh "Reliability" in Table 8 - Classification of OWL-S through the four aspects - speaks more precisely of "Service-Reliability". This Request Characteristic allows apparently a Service Requester to look for a Service Offer supporting a certain method for checking of the reliability of a service. The Char Properties of this characteristic could verify whether the Service Offer fulfils some concretely demanded parameters of the supported method. The semantic description of an RCh in Table 34 relates the requested services with a user group: "∀Can_be_used_by.Administrator". Such an association for a Request Characteristic is also principally illustrated in Fig. 28. This means that an ACTAS Administrator once introduced this characteristic, in order to enable users having an ontological defined administrator role to look for services supporting the intended reliability method. In fact, the quantifier "∀" might be sensed as too restrictive here, as it means that the selected services can only be used by a user group, in which each member is reduced to the administrator role. The existence quantifier ("∃") would only demand that the administrator role must be one of the roles of the members of the user groups (cf. section Table 5 - Constructors of DL). Fig. 38 and Fig. 28 also introduce a General Characteristic in the context of the nfp reliability, which leads to the extended usability of Semantic Characteristics as "building blocks" considered in the next section.

14.2.2 Extended usability of Semantic Characteristics

In the System Environment (chapter 8), it was introduced as one goal of an ACTAS Administrator to create and manage ontological repositories with Semantic Characteristics as instances. This changes the view on Semantic Characteristics as commonly agreed concepts classifying services as instances (as discussed in the last section) towards Semantic Characteristics as instances of ontological repositories themselves. The Service Designer (could be the Service Administrator mentioned as a user role in the description of the System Environment (cf. chapter 8)) can select semantically fitting Semantic Characteristics from these repositories through concepts, which reflect the used criteria in the semantic description. The found Semantic Characteristics can be used as "building blocks" for the Service Descriptions, in order to have semantically correct interfaces using the intended algorithms for matching and checking of constraints. The schemas of these ontological repositories have also to incorporate that properties of Semantic Characteristics can be "linked" through Exchange Constraints (described in section 10.4). One way of incorporation in the ontology is to relate the linked characteristic instances. An example of such kind of relations is illustrated through the "Works with" edges in Fig. 28, which relates instances of Compatibility Characteristics with instances of General Characteristics. (The figure obviously integrates both views on Semantic Characteristics.).

The tables in the appendix declare two Semantic Characteristics in the context of service reliability: a General Characteristic (equation (22-2)) and a Request Characteristic (equation (22-20)) both called "Reliability". However, the semantic description contains different concepts: the Compatibility Characteristic speaks of reliability on service level and the General

Characteristic on the level of a Composite Service. These characteristics could fit to the Compatibility and General Characteristic related through a "Works with" edge in Fig. 28 ($\mathsf{Works}_{\mathsf{with}}(Reliability, Reliablity)$). An explanation could be that the Request/Compatibility Characteristic "Reliability" ensures that a service supports a reliability criterion. When a service demands the composition of several services with this criterion, the service could use the General Service "Reliability", in order to calculate and check enhanced constraints. An adaptable Exchange Constraint, which collects certain values from the Char Properties of the Compatibility Characteristics, and which correlates them with Char Values of the General Characteristic, could be already included in the char-environment (cf. Definition 4) of the General Characteristic. The Service Designer could use the $\mathsf{Works}_{\mathsf{with}}$-relation, in order to find the Semantic Characteristics working with the General Characteristics or with the Compatibility Characteristic, respectively. In the latter case, DL offers the inverse use of the relation (cf. Table 5). For instance the equation ($\forall \mathrm{Works}_{\mathrm{with}}.\{Reliability\}$) gives back all Compatibility Characteristics, which only work with the General Characteristic "Reliability". On the other hand, the equation ($\exists \mathrm{Works}^{-1}_{\mathrm{with}}.\{Reliability\}$) returns a set of the General Characteristics, which at least work the Compatibility Characteristic "Reliability".

The hypothesis of ACTAS implies that the discovery and composition become more controllable with Semantic Characteristics. One aspect of services supports this vision directly: the trading aspect (4th aspect, phase 2). One can assume the introduction of a criterion, i.e. a concept in an ontology for the second phase of the life cycle ($\mathrm{Ontology}_{\mathrm{trading}}$), which is standing for a certain kind of trading. A Trader Agent of ACTAS responsible for this kind of trading will react when it finds a Compatibility Characteristic related with this criterion in a Trading Request (TRe, cf. section 11.2). The criterion can be used for the inclusion of the trading environments of other approaches. In this case, the TrA acts as a gateway to this trading environment: The information of the TRe is mediated to information necessary for the access of the external trading environment. With this information the Trader Agent performs a trading request with the external trading environment. When a matching service is discovered, the TrA could work as a FA for the found services.

The properties of a Compatibility Characteristic (CCh_Property in Definition 4) hold Property_Merge_Class descriptions, i.e. algorithms for the checking of compatibility constraints (so-called Merge Constraints). Characteristics could be associated with concepts selecting specific matching and mediation algorithms. Examples for such matching algorithms are OWL-MX [KluFri et al.2009] or WSMO-MX [KluKau2009], which include IR algorithms. WSMO supports mediation algorithms between services and between goal and service. Service Grounding and the Deployment may need a negotiation on the used Web Service standards. Some services use WS-I[7] for this purpose. Thus, it might be important for the pre-selection of services that they support such kind of negotiation. ACTAS could offer fitting characteristics.

[7] OASIS, 14.01.2011

14.3 Service Request

As shown in Definition 19, the Service Request is, despite its similarity in its data structure to a Service Template, an entity, which holds various kinds of information. Firstly, it consists of several Client Request Records with Request Ports as interfaces, since a Service Request in ACTAS poses a request on behalf of several Service Clients. Secondly, a Service Request possesses a Common Part, which allows the setting of constraints for the whole Composite Service (cf. Definition 19). In the Service Composition, a Client Request Record is used to generate an Actor Service Offer (ASO) (cf. C-Model, section 13.3). The Request Characteristics describe a Request Port in the ASO.

$$\begin{aligned}
&\text{SRe}_{\text{DFC}} = \left(\text{SRe}^{\text{ID}}, \text{ReA}^{\text{Ref}}, \text{SRe-Common}, \text{Client-Request}^{\text{Set}}\right)\\
&\text{Client-Request}_1 = \left(\text{RM}_1^{\text{ID}}, \text{PA}_{\text{A}}^{\text{Ref}}, \text{AST}_{\text{A}}^{\text{Ref}}, \text{RP}_1^{\text{Set}}, \{\quad\}, \text{SRe-Env}_1\right)\\
&\text{Client-Request}_2 = \left(\text{RM}_2^{\text{ID}}, \text{PA}_{\text{A}}^{\text{Ref}}, \text{AST}_{\text{A}}^{\text{Ref}}, \text{RP}_2^{\text{Set}}, \{\quad\}, \text{SRe-Env}_2\right)\\
&\text{Client-Request}_3 = \left(\text{RM}_3^{\text{ID}}, \text{PA}_{\text{A}}^{\text{Ref}}, \text{AST}_{\text{A}}^{\text{Ref}}, \text{RP}_3^{\text{Set}}, \{\quad\}, \text{SRe-Env}_3\right)\\
&\text{RP}_{1,1} = \left((\text{RM}_1, \text{RP}_1)^{\text{ID}}, \{\text{HotelBooking, FlightBooking, CarHiring}\}, \text{SRe-Env}_3\right)\\
&\text{RP}_{2,1} = \left((\text{RM}_2, \text{RP}_1)^{\text{ID}}, \{\text{CompleteBooking}\}, \text{SRe-Env}_4\right)\\
&\text{RP}_{3,1} = \left((\text{RM}_3, \text{RP}_1)^{\text{ID}}, \{\text{HotelBooking}\}, \text{SRe-Env}_5\right)\\
&\text{RP}_{3,2} = \left((\text{RM}_3, \text{RP}_2)^{\text{ID}}, \{\text{FlightBooking}\}, \text{SRe-Env}_6\right)\\
&\text{RP}_{3,3} = \left((\text{RM}_3, \text{RP}_3)^{\text{ID}}, \{\text{CarHiring}\}, \text{SRe-Env}_7\right)
\end{aligned}$$

Code 8 - Service Request for Alternative Client Requests

In Example 16, Alternative Client Requests were discussed, which would allow a Service Client A to ask for the booking of a flight, the booking of a flight, and the hiring of a car in different ways. In the context of this example, it was sketched, that Alternative Client Requests would get recognisable in a Service Request through the referencing of the same Personal Agent (if the application environment does not include Personal Agents, an identification of the Service Client should be used instead). In Code 8 - Service Request for Alternative Client Requests, Alternative Client Requests for the example are given. The resulting ASO is shown in Fig. 40.

An ASO is created for every Service Client stated in a Service Request. For Alternative Client Requests, it is only one potential Service Client in accordance with its definition. Therefore, all three Client Requests in the code appear as Request Modes in one Actor Service Offer in the figure. The existing duality between Service Offer and Actor Service Offer becomes apparent. Like the Service Modes of a Service Offer are selected due to the rule of principal during the Composition Process, so a Request Mode must be selected by the Composition Agent for the initialisation of the extended Composite Structure (CompSt-plus) (cf. section 13.3).

The first Request Mode (RM_1) has only one Request Port holding three Compatibility Characteristics for the mentioned services. In the appendix, these characteristics are introduced with a semantic description relating them to their domain and a Service Design, which demands WSMO goals in a Service Request and WSMO services in the Service Modes of principally compatible Service Offers. Thus, every Service Provider, who is offering these three services of different kinds together, could publish a Service Template containing these three Compatibility Characteristics.

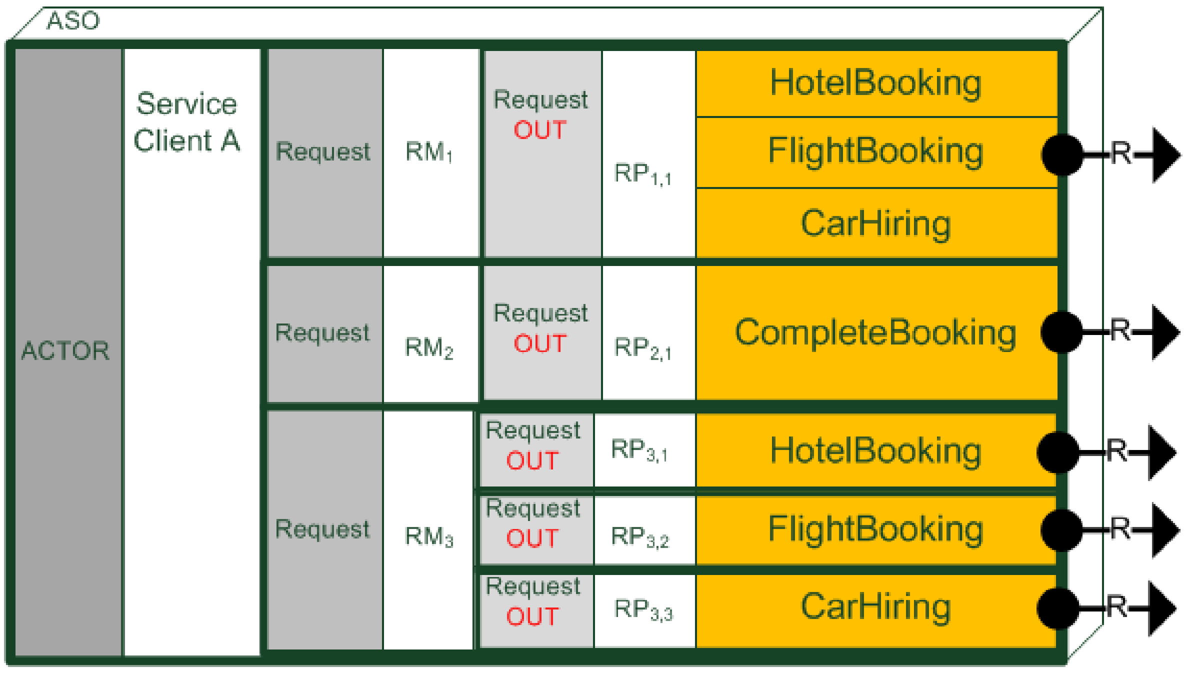


Fig. 40 - Actor Service Offer (ASO) for Alternative Client Requests

Probably, the same Service Providers could answer the alternative Request Mode RM_2, since it uses a Compatibility Characteristic "CompleteBooking", which allows several different kinds of bookings including the ones needed in this case. In a WSMO-based booking environment, a Service Requester would use such a booking service, in order to get the three services composed in a proper way. Alternatively, the services could be discovered separately and composed through the discovery environment. This happens with the third Request Mode RM_3. The Compatibility Characteristics stand in different Request Ports. In the WSMO-based environment, three separate requests would be necessary.

Like in a WSMO-based environment, the "CompleteBooking" Compatibility Characteristic would demand the existence of an extra booking service. However, the booking itself could become quite flexible, since it will be possible to offer only a subset of the requested services, when the originally requested services cannot be composed. In comparison of this Compatibility Characteristic with the combination of simpler Semantic Characteristics in the Request Port of the first Request Mode, it can be recorded that it might be easier to discover services with simple Semantic Characteristics. However, their combination does not necessary request the same services as a Semantic Characteristic, which combines in its semantic descriptions the semantic descriptions of the simpler ones.

14.4 Composition Process

The Request Agent (ReA) is part of the Application Environment and it is the interface to the environment of ACTAS. The ReA will create the Composition Agent (CoA) with a behaviour adapted to the application environment. Therefore, the CoA can be used, in order to integrate established algorithms for Service Composition. In this section, an overview about different ways of doing the Composition Process is discussed. The terms in Table 22 are used in the description. Four cases are differentiated through the Composition Graph (cf. Definition 27):

- 1st case: the resulting Composition Graph is a tree
- 2nd case: the resulting Composition Graph does not contain a loop
- 3rd case: the resulting Composition Graph is not restricted
- 4th case: several edges between vertices are allowed

The Composition Algorithm starts with the Open Ports of the Actor Service Offers (ASO) ($S_C = ASO^{Set}, \forall aso \in ASO^{Set}: Open = \bigcup \wp_{open}(aso)$) created of every Client Service Request of the Service Request (cf. section 13.3). The primary goal of the Composition Algorithm is the looking for compatible services for the Open Service Ports (Principal Compatibility in Definition 11 and equation (14-1)). The goal is an empty set of Open Ports ($Open = \emptyset$), i.e. for all Open Ports a compatible Service Port was found and composed. The result is the composition graph $G = (S_C, M_C)$ (cf. Definition 27).

Principal compatibility in formula:

$$\begin{aligned} &\exists so_1, so_2 \in SO, \exists s_1, s_2 \in S, \exists p_1, p_2 \in P: \\ &s_1 \in \mathcal{s}(so_1) \wedge s_2 \in \mathcal{s}(so_2) \wedge p_1 \in \wp(s_1) \wedge p_2 \in \wp(s_2) \wedge c^{Ref}(p_1) = c^{Ref}(p_2) \end{aligned} \tag{14-1}$$

The data of the Open Port contains a field called "depth" (cf. Definition 27). Its definition keeps in mind that the Composition Process starts with a Service Request and that an ASO is based on a Client Request of a Service Request. The introduced algorithm of the Composition Process in the C-Model description uses the field "depth", in order to sort the set of Open Ports in an ascending order. The Open Port with the smallest path length was chosen for the next Service Discovery and Service Composition step. However, only Out Ports or non-directed Ports were selected, since IN Ports only offer a service.

A more elaborated way to deal with the set of Open Ports is the splitting up of the set $Open$ in the subsets for IN Ports ($Open_{IN}$), OUT Ports ($Open_{OUT}$), and non-directed Open Ports ($Open_{non}$). In this way, the OUT Ports and non-directed Open Ports can be easier selected and managed. In the following paragraphs some cases for the Composition Process are covered.

1st case) the resulting Composition Graph is a tree

The resulting graph will become a tree, when the selected Service Modes (S_C) may have only one IN Port ($\forall sm \in S_C: |\imath(sm)| = 1$), but there is no restriction for number of the OUT Ports. A Composition Agent, which follows an algorithm with this rule, could look effectively for information retrieval services or services of the Cloud Computing, which distribute the computation load on several services. The Compatibility Characteristics of these services will help to get just these services.

2nd case) the resulting Composition Graph does not contain a loop

The CoA wants to avoid loops in the composition graph, therefore it follows the following rule: the OUT Ports of a selected Service Mode sm_x will only become members of the Open Ports ($Open_{OUT} = Open_{OUT} \cup \sigma(sm_x)$), when all IN Ports of sm_x have been composed ($Open_{IN} \cap \imath(sm_x) = \emptyset$).

> Proof: It shall be assumed that a service (vertex) sm_y exists in the set of selected Service Modes S_C of G assembled with the above stated rule and sm_y shall have a loop starting at one of its OUT Ports.
> Length of the loop is 1: This cannot be because the OUT Ports of sm_y would be only considered for composition, when all IN Ports are composed.
> Length of the loop is greater than 1: When such a loop occurs, the vertex sm_y must have got a path reaching a vertex sm_a, which had already a path with an IN Port of sm_y. Following the rule, there will be always for every vertex an earlier time t, when all IN Ports are composed, and a time t', when an OUT Port was composed (t < t'). For the vertices and their OUT Ports on the assumed path may these times ($t_y < t'_y$) and ($t_a < t'_a$), respectively. Since sm_y was assumedly connected with sm_a it must be valid ($t'_y < t_a$), but sm_a shall also have a path to sm_y, i.e. ($t'_a < t_y$), and this leads to non-resolvable conflict.
> Therefore no loop can exist. <q.e.d.>

The avoidance of a loop inside of the graph ensures that no service relies on the other one. ACTAS does not do a Service Composition on process level; these issues could be clarified transparently through Merge Constraints. However, the directed Service Composition between Client and Server Ports normally express a reliance relation between the services, i.e. a service is likely to need the services requested with the OUT ports, in order to provide the services offered at the IN ports. A supply chain might want to avoid such kind of reliance for the prevention of resource conflicts.

3rd case) only one composition between two selected Service Modes

In the third considered case, the rules of the (composition) graph shall still be observed, i.e. there shall be maximum one edge between two vertices. Loops are allowed. In a supply chain or other e-business scenario, it can occur that a service which relies on other services can also offer a service, which is of interest for the services; it relied on in the first place. It might be even important for the supply chain that this loop in the composition graph exist. In the last case, the loop was excluded, in order to avoid constraints of resources. An extra Option-Slot could ensure that the loop is established by the Composition Agent.

The application environment determines the policies for the composition of a CoA. A CoA for the composition of communication services will likely try to achieve a connected graph keeping the mentioned "depth" value as low as possible (cf. Definition 24). The value of the "depth" field could be also used, in order to decide about the fail of a Service Composition, when it exceeds a given threshold. An example for a CoA, which is interested in a connected composition graph, is discussed in the next paragraphs in the context of MCU-Conferences.

A MCU-Conference enables a flexible number of Service Clients having a communication with the help of a Multi Communication Unit (MCU). In the past, a MCU has been often a technical facility installed in a special room, which had to be booked, in order to be used. Nowadays, applications like Skype offer similar solutions with an ad-hoc access. In the ideal System Environment of ACTAS, the Personal Agent could verify, if and when a user is available for the use of conference applications like Skype.

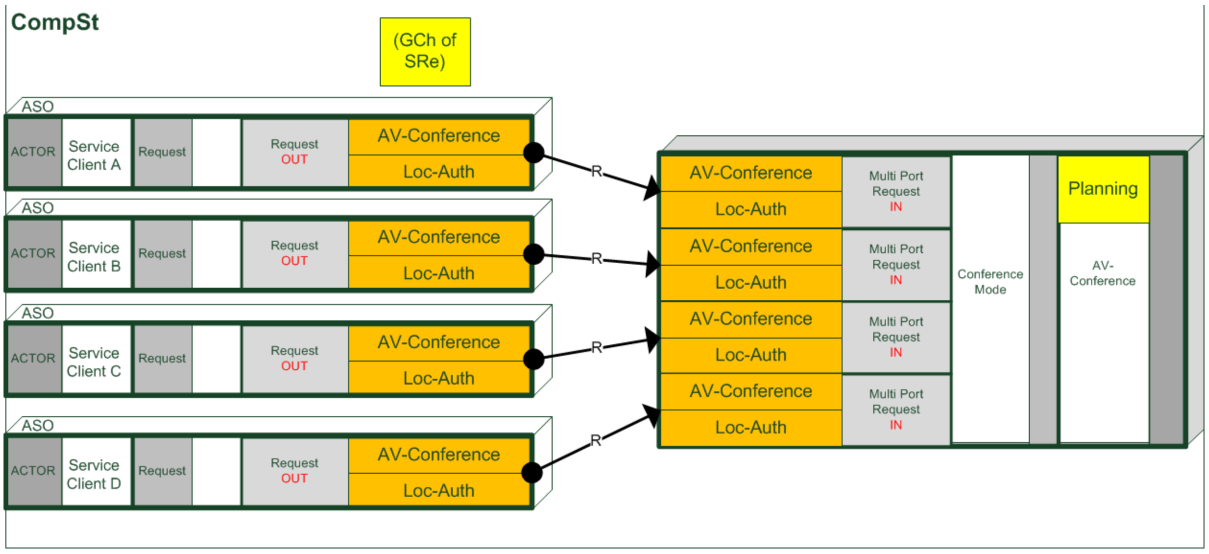


Fig. 41 - Composite Structure (CompSt) with use of Multi Ports

In the terms of ACTAS, a MCU is represented through a Service Mode of a Service Offer, which offers an adaptive number of Service Ports depending on the number of Service Clients and their sub-graphs of services, which shall be composed with the Service Offer of the MCU. The selection of the MCU Service Mode for the Composite Structure means, that the sub-graphs, which started with the ASOs built of the Client Requests, get connected. In Fig. 41 - Composite Structure (CompSt) with use of Multi Ports, an example of a Composite Service with a MCU is shown. The Multi Port offers an Audio-Video-Conference through the Request Characteristic "AV-Conference" (cf. equation (22-13)). Additionally, the Request Characteristic "Loc-Auth" (cf. equation (22-16)) is used, in order to clarify that the MCU is reachable at a time slot and the costumer has an authorisation for the using of the MCU. The General Characteristic "Planning" could verify the planning of the time slots of the Service Clients. In this way, even a conference with Skype could get planned as long as the application environment, which generated the Service Request for the AV Conference in the first place, offers a user interface in this direction.

In section 10.2, the Option-Slots for the control of the behaviour of the Composition Process were portrayed. A "multi-port" Option-Slot was introduced for the realisation of a Multi Port in Table 13. The environment description (ST-Env in Definition 8) of a Service Port declared as a Multi Port will contain this Option-Slot. It should at least specify the number of allowed replications of the Service Port. The MCU might not demand that all possible Service Ports are used. Thus, the FA could offer a Service Template for the MCU, which holds a Service Mode with the number of Service Ports, which have to be composed as IN Ports in order to be able to use the MCU. One of these Service Ports could be additionally declared as a Multi Port. A Service Offer based on a Service Template with a Multi Port can automatically create a new Service Port with the same information if useful in the Composition Process. Every Service Port of a MCU Service Offer is based on the same basic information set through the Value Constraints. In this way, an additional Service Client could use the MCU-facility.

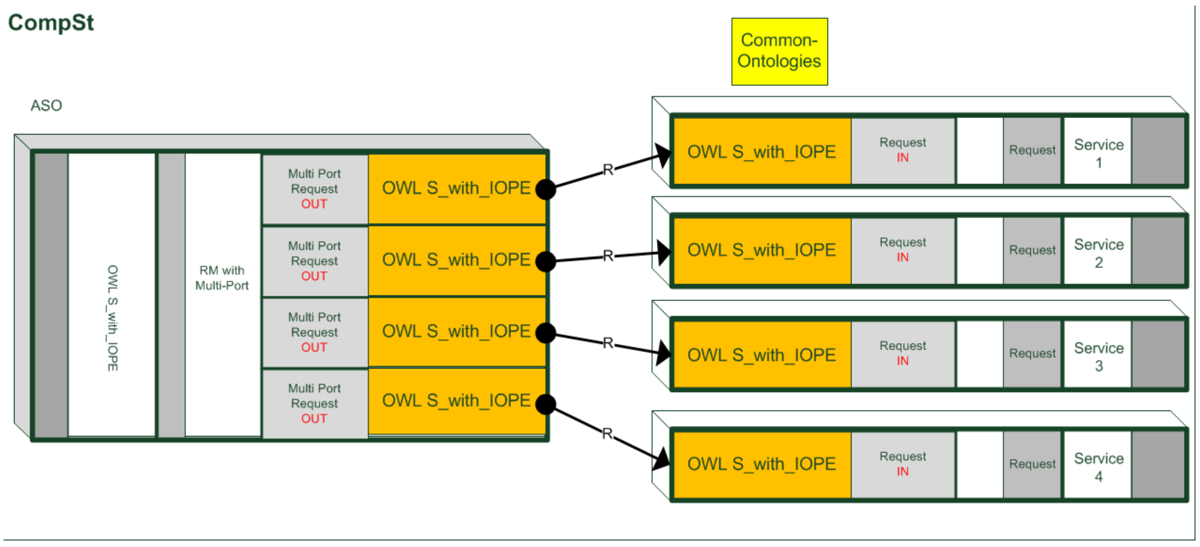


Fig. 42 - Multi Port in an ASO for OWL-S Service Composition

A lot of publications like [ÇelElç2008] exist, which consider the composition of services based on the IOPE capability description of SWS. OWL-S is mainly using this kind of capability description. Many of the approaches, which could be adapted for the behaviour of the CoA, use the input and output parameters given in the IOPE capabilities, in order to compose services. Principally, they all try to produce the wished output parameters with the provided input parameter. If one Component Service cannot produce all output parameters, a Service Discovery for another Component Service will be started for the left output parameters. The Component Service might follow the same principle on the next level, in order to produce its output parameters. A complex Composite Service can be the result.

This scenario can become another application of the Multi Port in ACTAS, as shown in Fig. 42. The OUT Ports of an ASO were declared as Multi Ports. This is the inverse application of a Multi Port in comparison to the previously discussed application, where an IN Port was replicated. The Service/Request Ports in the current example contain OWL-S Service Descriptions wrapped in the Compatibility Characteristics "OWL-S_with_IOPE", which were

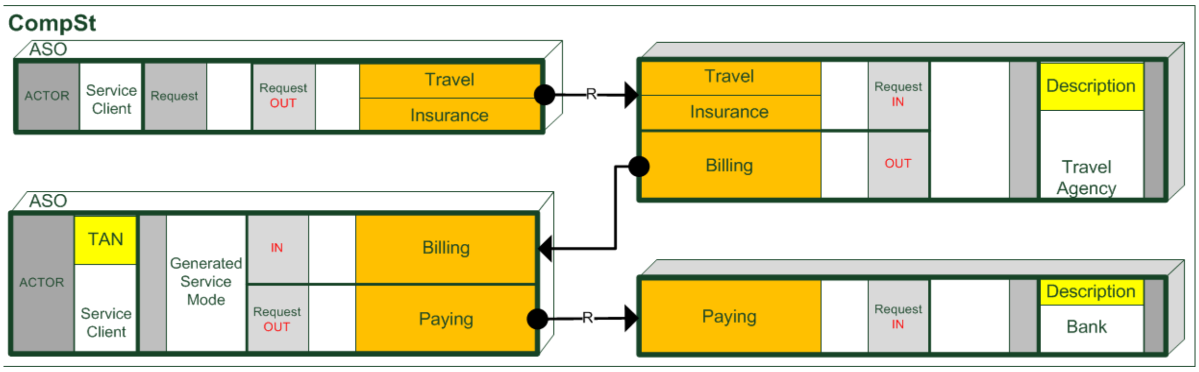


Fig. 43 - Business Service with principal compatibility

ealier covered. The Merge Property Class associated with the Char Property holding the Service Description could give access to the matching results of the input and output parameter for a Service Composition. Following, the mentioned algorithms and knowing the restrictions given through the "multi-port" Option-Slot, the CoA can decide to create a new Request Port. Fig. 42 shows a possible result of the CompSt. The information in the OWL-S Service Descriptions helps to perform the subsequent phases of the life cycle.

The discussion above showed that a Service Mode can have Service Compositions with several other Service Modes. However, there is only one composition between two distinct Service Modes possible in the current case (3rd case). The Multi Port makes a Service Mode adaptable. Therefore, the possibilities for the dynamic adaptation of the parts of the Service Description of ACTAS shall be shortly discussed in the following.

The set of Compatibility Characteristics in the Service/Request Ports decide the principal compatibility of services in ACTAS. A Trading Request (TRe, cf. Definition 20) contains a given set of Compatibility Characteristics, which should not be changed, because the Composition Process relies on them. If a Facility Agent or Trader Agent recognizes through the semantic descriptions of the Compatibility Characteristics that it could offer a service fulfilling the principal compatibility then the agent will react and extend the Service Template accordingly if necessary. Such a dynamic extension of the ST done by the agent will be implemented in future versions of ACTAS.

A further consideration would be the extensions of the rules of principal compatibility. Similar to the matching ideas for IOPE matching in SWS, a Service Port offering less or more Compatibility Characteristics than the demanded ones, could be still declared as a kind of principally compatible. However, these ideas shall not be applied, since the agreement on the categories is an essential principle of ACTAS.

The dynamic extension of the number of Service Ports of a Service Mode was answered with the Multi Port idea. However, it shall be only applied where the Service Designer set the "multi-port" Option-Slot (cf. Table 13 - Option-Slots of Service Ports). Two Service Ports realises a fixed directed or non-directed Service Composition between two Service Modes. In the State-of-the-Art, it was discussed that in a Business Service the role of the involved parties can change from a client to a provider and vice versa. In Fig. 43, this case is shown for the Service Client., who firstly requested a “Travel” service and combined this with a Compatibility Characteristic for an “Insurance” service. One Service Port does only support one direction of Service Composition. Nevertheless, the Service Provider felt obviously the need to settle the billing with the costumer as early as the Service Discovery of ACTAS. At this time point the Personal Agent of the costumer could come in the role of a Facility Agent. The Personal Agent could manage an Actor Service Template (AST) similar like a Facility Agent its ST. In this way, a professional costumer could provide a Service Mode for the billing. This Service Mode is then selected for the Composite Structure.

4th case) Several Service Compositions between two Service Modes

The fourth case leaves the highest degree of freedom for the Service Composition in the Composition Process. Even the definition of the Composition Graph in Definition 27 would have to be adapted, in order to accept several edges between two vertices. This adaptation is done through the introduction of the triple (V, E; I), V as the set of vertices, E as a set of entities for the edges, and I as an incidence relation (I ⊆ V×E). An application shall be sketched. The link between two Technical Services can be split in several single links. For instance the transport of gas or water between two sides can be done in several single pipelines (cf. Fig. 44). The Composition Process could apply methods offered by the used Merge Property Class of the Compatibility Characteristic “Pipeline”, in order to find out the used capacities of the pipelines and the wished amount of transported gas. The different views on the Merge Property Objects (cf. Definition 17) are certainly helpful for this purpose. Finally, the CoA will decide whether it adds a new pair of Service Ports with the Compatibility Characteristic “Pipeline” depending on the requested amount and the current resources. This idea could also involve the negotiation of agents. The Composition Process could integrate graph algorithms like the Ford-Fulkerson’s Algorithm for the maximum flow between two vertices, i.e. Service Modes for water or gas supply in the assumed scenario.

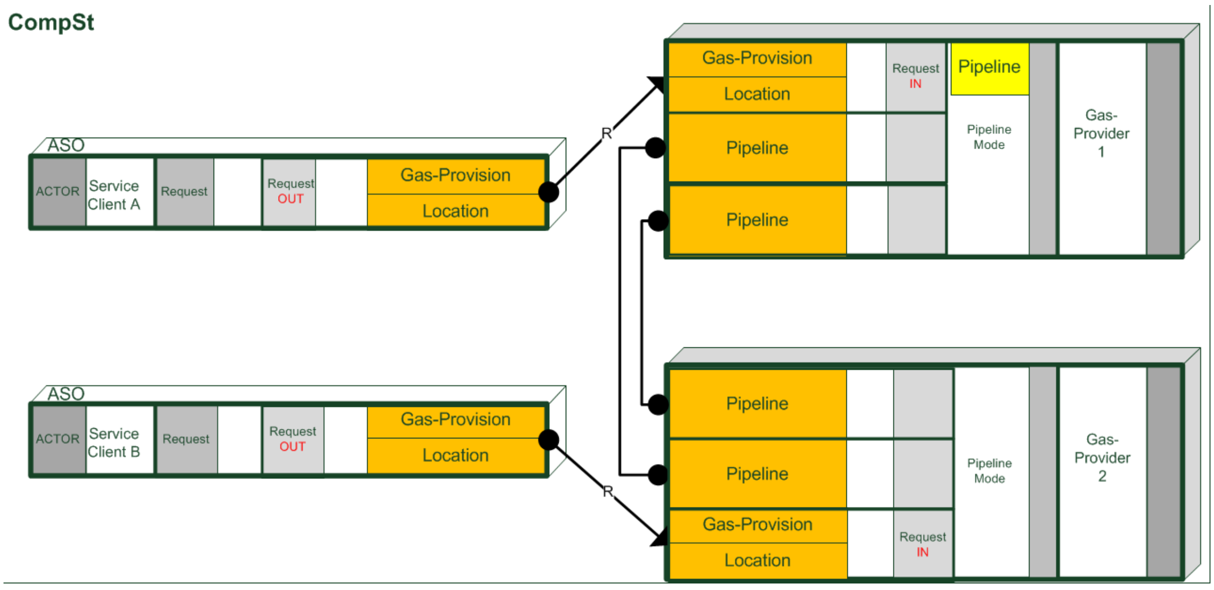


Fig. 44 - Several compositions between selected Service Modes

Term	Description
$G = (S_C, M_C)$	Composition Graph as introduced in Definition 27
$K \subset I_C \times O_C$	Set of directed edges in CompSt $\forall (i,o) \in K\colon \exists m \in M_C \wedge cl(m) = o \wedge se(m) = i$
$L \subset \wp_2(N_C)$	Set of non-directed edges in CompSt
Sets	
S_C	Set of all Selected Service Modes in CompSt
M_C	Set of all Merged Service Ports in CompSt
$Open_{IN}, Open_{OUT}, Open_{non}$	Sets of Open Ports in CompSt-plus (with direction IN, OUT, non)
I_C, O_C, N_C	Set of Service Ports in CompSt-plus (with direction IN, OUT, non)
S	Set of all Service Modes
P	Set of all Service Ports
SO	Unified Set of all Service Offers and Actor Service Offers
PO	Set of all Property Objects, the handles to the implementation instances
Functions	
$\mathscr{p}: S \to \wp(P)$, $\mathscr{p}_{open}: S \to \wp(P)$	Getting Service Ports of a Service Mode Getting Open Ports of a Service Mode
$\mathscr{i}: S \to \wp(P)$	Getting IN Ports of a Service Mode
$\mathscr{o}: S \to \wp(P)$	Getting OUT Ports of a Service Mode

Term	Description
$\mathscr{n}: S \rightarrow \wp(P)$	Getting non-directed Ports of a Service Mode
$c: P \rightarrow \wp(CCh^{Set})$ $c^{Ref}: P \rightarrow \wp((CCh^{Ref})^{Set})$	Getting the set (of references) of Compatibility Characteristics from a Service Port
$\mathscr{g}: S \rightarrow \wp(GCh^{Set})$	Getting the set of General Characteristics from a Service Port
$\mathscr{s}: SO \rightarrow \wp(S)$	Getting the set of Service Modes from a (Actor) Service Offer
$cl: M_C \rightarrow P$	Getting the Client/IN Server Port of a Merged Service Port
$se: M_C \rightarrow P$	Getting the Server/OUT Server Port of a Merged Service Port
$me_{prop}: M_C \rightarrow MeProperty^{Set}$	Getting the MeProperty from a Merged Service Port
$cl_{view}: MeProperty \rightarrow PO$	Getting the Client/IN Server Port of a Merged Service Port
$se_{view}: MeProperty \rightarrow PO$	Getting the Server/OUT Server Port of a Merged Service Port
$me_{view}: MeProperty \rightarrow PO$	Getting the Merged Port

Table 22 - Terms for the description of the Composition Process

15 Case Study 1: Technical Services with translation

Convergence of data and telecommunication networks leading to a coherent and transparent technology promises better services and improved quality. The liberalization of the telecommunication market allows the users the choosing of different operators for various services. The result is a portability of services with respect to users' needs enabling often already for instance the transparent combination of data, voice, and video. As Cloud Computing shows, Technical Services are increasingly combined with an elaborated prizing schema. Thus, the aspects of different domains become combined and the search for a telecommunication Service Providers involves the comparison of prizes. Currently, proprietary solutions dominate the market in Cloud Computing leading to the previously mentioned Lock-in-Effect. However, the pressure of the market will lead to autonomic Service Discovery and Composition. The transparency of the technical side towards the Service Clients was discussed in the State-of-the-

Art. From this point of view, Technical Services distinguish between B2C and B2B interfaces. The latter interfaces used for the realization of the Technical Service itself.

15.1 The ACTAS Administrator

ACTAS supports the distinction of the two kinds of interfaces through the introduction of a special kind of Compatibility Characteristic, the Request Characteristic used for the description of the interface towards the Service Requests. On the one hand, an ACTAS Administrator could rely on the existing networks and would not introduce specific Compatibility Characteristics for the description of the technical interfaces. On the other hand, just the existing plurality of Technical Services, partly intentionally increased as stated, could make a matching of service parameters on the level of ACTAS interesting for Technical Services.

Thus, the ACTAS Administrator could introduce several Semantic Characteristics wrapping such parameters in a specific semantic context. For the description of the internal interfaces, he could publish Compatibility Characteristics like "Phone" for a telecommunication service as given between telephone switchboards or exchange facilities. The audio-video standard H.323[8] for IP-based networks could lead to the introduction of another Compatibility Characteristic, named "H.323", wrapping Char Properties, which help to clarify parameters of H.323-based connections.

Already Example 1 showed a communication service between two Service Clients having communication facilities connected to different networks (telephone and H.323). In Autonomic Service Oriented Computing, this fact shall stay transparent for them; especially the needed network gateway is concealed. For the Service Client interface, the ACTAS Administrator can introduce Request Characteristics describing the interface for the Service Clients. The Service Clients do not describe their wished services in technical terms and it is not likely that they are firm in the WSML language. Therefore, the ACTAS Administrator decides to publish a bunch of Request Characteristics having Char Properties for a simple description: "General-Com", "AV-Com", "Audio-Com", and "Written-Com". The first one should accept communication with any combination of data, audio (includes simplex or higher quality), video, or voice. "AV-Com" supports audio and video combinations. Blackboard, chat, or e-mail communication should be covered with "Written-Com".

The ideas of the ACTAS Administrator do not stop here. He realises, that Exchange Constraints will be needed for the data mediation. ACTAS speaks in this context often of "translation", because it works with Property Objects (cf. C-Model), in order to access the information hold in the implementation instances. Due to the declarative environment, the Property Objects used in the Exchange Constraints have to be "borrowed", their information mediated, and then returned to the information of the Composite Service.

[8] IEC, 2007

The ACTAS Administrator could consign the task of writing the Exchange Constraint to the Service Designer. However, this would make the handling of ACTAS clumsy, since the Service Designer would have to look up matching Exchange Property Classes on his own every time, when he wanted to use the characteristics. ACTAS offers the possibility to introduce Semantic Characteristics as "building blocks" for such translation tasks through pre-defined Exchange Constraints. Mostly General Characteristics will be used for such a task. The General Characteristics can be placed in the Service Mode or the Common Part of the Service Description, i.e. the Service Template. The ACTAS Administrator introduces General Characteristics for the translation task like the following ones: "Audio-Phone", "Audio-H.323", and "AV-H.323". These General Characteristics could have several Exchange Constraints, in order to mediate from the information of the user interface to the information of the technical interfaces.

The "ExchangeProperties" term (cf. Definition 18) of a pre-defined Exchange Constraint can be empty or contain the names of Property Classes, which can be associated with the Exchange Names. In the "view" field, it can specify the preferred view, in order to borrow a Property Object of that Property Class from a Merge Property Object (cf. section Definition 18 (Ex-Co in C-Model) Definition 15, and Definition 17). An adaptation of this pre-defined Exchange Constraint through the "exchangeProperties" Option-Slot was illustrated in Example 15, which showed a principal Service Mode holding a General Characteristic "Audio-Phone" with the pre-defined Exchange Constraints, which in its adapted version translated values of the Semantic Characteristic "Audio-Com" into values of the Semantic Characteristic "Phone".

The principle of translation with an Exchange Constraint inside of a General Characteristic can be extended through the use of so-called **Translation Offer**s. In section 14.2, it was already motivated that the vision of WSMO is based on mediators doing the mediation between its components through centralized algorithms, i.e. external algorithms accessed through URLs. The Translation Offers of ACTAS extend this vision. They set the mediation algorithms and other centralized algorithms in the semantic context described through the Semantic Characteristics and their definitions. As previously discussed, the mediators of WSMO can be used in the constraints of ACTAS. The Value Constraints setting a goal or service component could integrate the mediators defined for these components in their term "usesMediatior" (cf. [FeKeZa2008]). For instance the component goal uses the ooMediator for data-to-data mediation and a ggMediatior, in order to describe some extensions in comparison to another goal. The Merge Constraint for a WSMO description could use besides the matching algorithm the mediators allowed between goal and service. Finally, the Exchange Constraint could take advantage of any mediator, in order to mediate between the WSMO elements. In ACTAS, the Exchange Constraints extend the mediation to values not defined in WSMO.

A Translation Offer offers the mediation mainly based on Exchange Constraints to the service world of ACTAS. It allows for instance the translation from one user interface (Request Characteristic) to another one. Thus, service can be provided without the complicated Service

Description of SWS, but with restricted and adapted describing options for the user application. A Translation Offer can translate this user interface to the SWS interface, in order to use their methods. In the following paragraphs the use of Translation Offers in the context of this case study is covered. ACTAS speaks of "translation" instead of "mediation" due to the just discussed extensions to the vision of WSMO.

15.2 Translation Offer

In Fig. 45, an example of a Translation Offer is shown. It translates information between the Request Characteristics "AV-Com", "Audio-Com", or "Written-Com" and the Request Characteristic "General-Com". These are the Request Characteristics, which were assumedly introduced by the ACTAS Administrator earlier on.

The advantage of these Translation Offers is that the Service Designer has not any longer to think himself about the creation of Service Modes with General Characteristics for the translation of different interfaces, his Service Offer could support. He can rely on the existence of

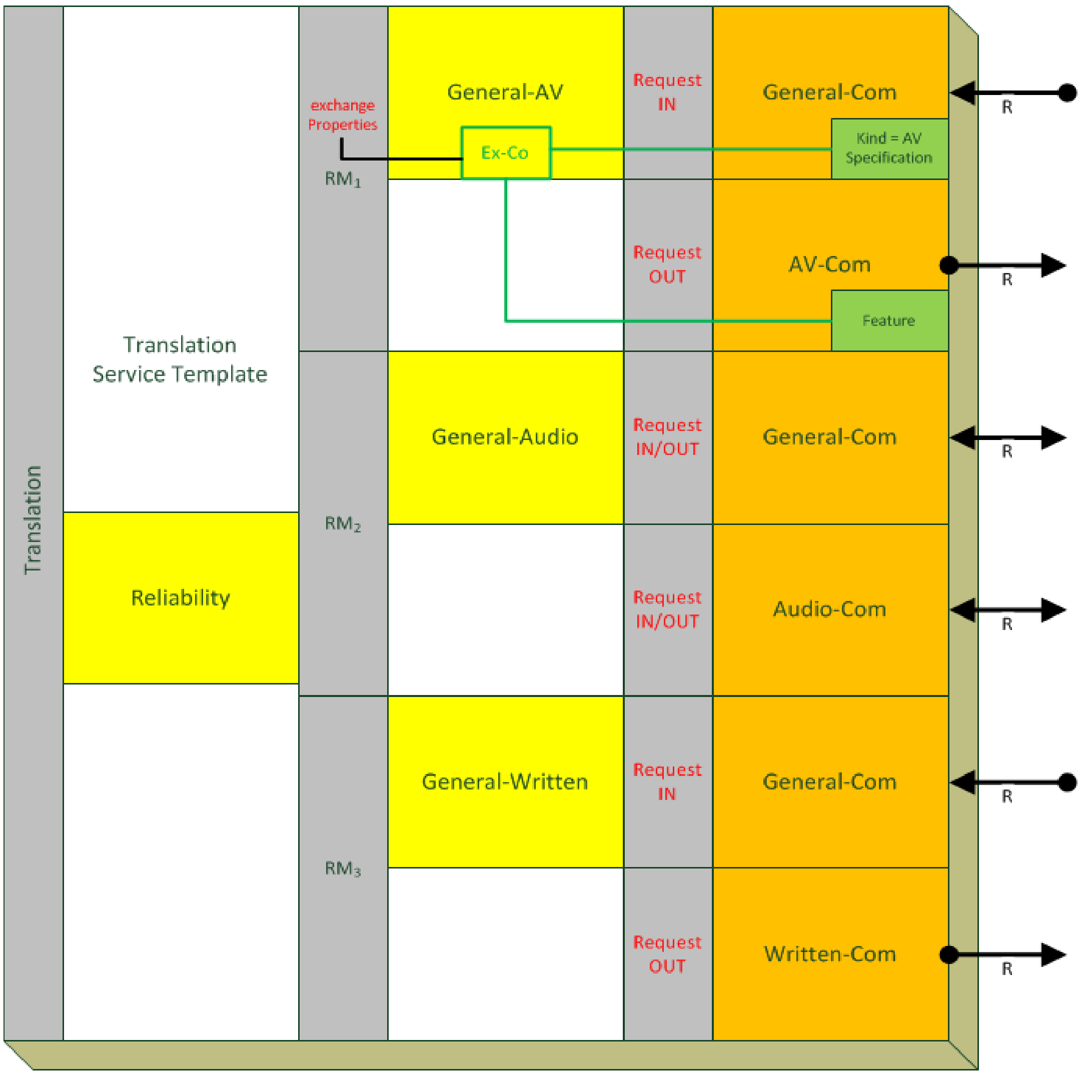


Fig. 45 - Example of a Translation Offer of ACTAS

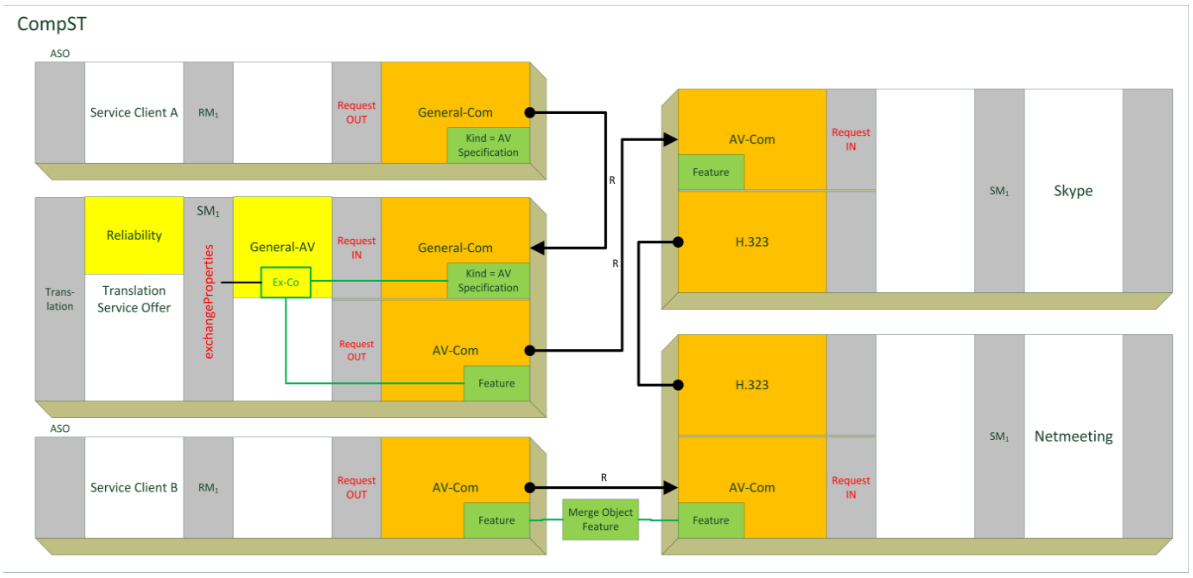


Fig. 46 - Audio-Video-Communication with Translation

Translation Offers. The Translation Offer in Fig. 45 uses the General Characteristics “General-AV”, “General-Audio”, and “General-Written”, in order to keep pre-defined Exchange Constraints for the mediation of the values of the enumerated Compatibility Characteristics. The figure also shows the effect of the “exchangeProperties” Option-Slot. Service Properties of the Compatibility Characteristics “General-Com” and “AV-Com” are associated with Exchange Names of the pre-defined Exchange Constraints.

Finally, the Service Designer can use the provided and published Semantic Characteristics. He defines Service Templates as discussed in Example 9, which consists of several Service Modes. The design of Service Descriptions with Semantic Characteristics as “building blocks” was sketched in Example 15, which portrayed the use of pre-defined Exchange Constraints for the General Characteristic “Audio-Phone”, providing a translation from the Request Characteristic “Audio-Com”, used in the user interface, to the Compatibility Characteristic “Phone”, used in the transparent technical interface previously discussed. The Service Modes of the Translation Offer provide a similar translation, but between user interfaces, i.e. it is dealing with Request Characteristics.

Example 20 **Audio-Video-Communication with Translation**

A Service Request looks on behalf of a Service Client A for a general way of communication (“General-Com”) and on behalf of a Service Client B for an audio-video-communication (“AV-Com”). In the Composition Process, it turns out that “General-Com” has to be translated, in order to find directly Service Offers for AV-Com. It is assumed; that otherwise the Composition Process could not find composition graphs with a satisfying path length (in short the “depth” value became too high every time so a backtracking was forced, cf. section 14.4). A resulting Composite Structure is shown in Fig. 46 - Audio-Video-Communication with Translation. It illustrates the used Exchange Constraint for the translation and the Merge Constraint on one of the concerned Service Properties.

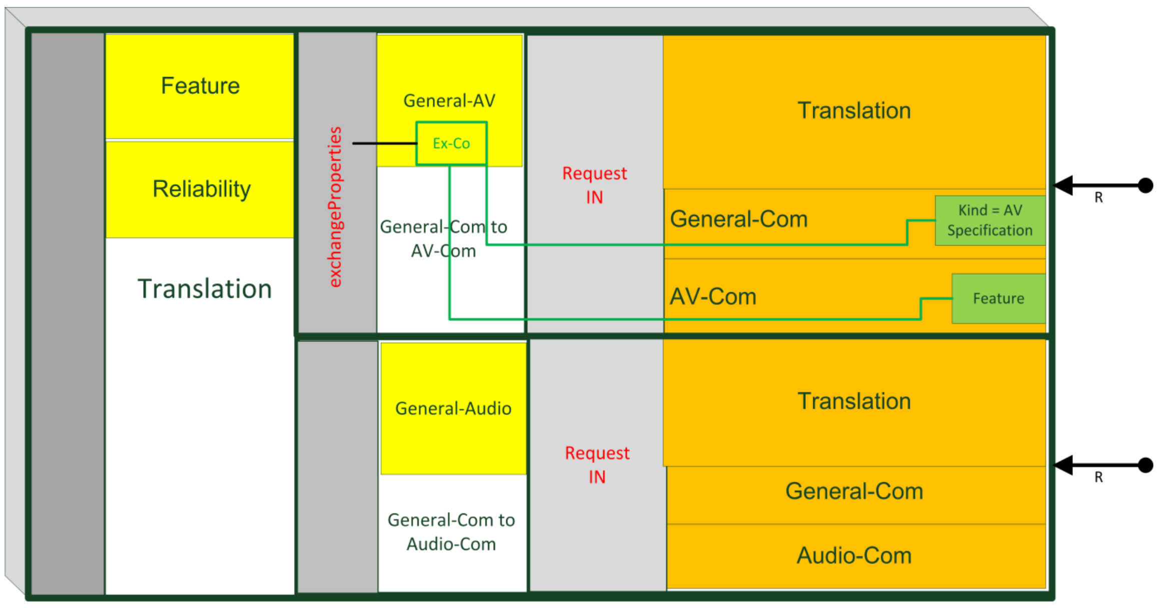


Fig. 47 - Alternative Example of Translation Offer

In Example 17, the start of a Service Composition of a telecommunication service is shown through Service Request, ASOs, and initialised Composite Structure. The Composition Process with the necessary use of a gateway in telecommunication is portrayed in Example 18. In Example 19, the resulting Composite Structure is given. The dual steps are shown for the audio-video-communication with translation in the subsequent example.

The Fig. 46 shows that the Service Mode RM_1 was chosen. (Remark: the identification of the Service Port shows that is actually even a Request Mode, since it only contains Request Ports, i.e. Request Characteristics, a translation of user interfaces.) The Translation Offer will have the "translation" Option-Slot (cf. Table 14), in order to ensure that Exchange Constraints for the translation are applied before the next Service Discovery step with a Trading Request (cf. C-Model, section 13.4 - Step 3: Service Discovery and Principal Compatibility). It is this Option-Slot, which indicates towards the CoA that it actually deals with a Translation Offer.

At closer inspection of the sketch Composition Process, one could complain that it is actually undetermined when the Translation Offer is actually discovered and applied. On the one hand, this is not necessarily a disadvantage, since a Service Offer might be discovered with the Compatibility Characteristic "General-Com", which integrates best into the Composite Service. On the other hand, special Trader Agents could exist for the publication of Translation Offers. When the Composition Agent early recognises that a translation is necessary, it will directly address these Trader Agents with its next Trading Request for e.g. "General-Com". The CoA might even introduce an Option-Slot, in order to the Trader Agent with the TRe, which Semantic Characteristics are wished as goal of the translation.

At the beginning of the evaluation, in chapter 14, the use of a criterion classified as belonging to the second phase, the Trading Phase, of the life cycle (4th aspect, phase 2) was discussed as a criterion for the semantic description of a characteristic. When such a criterion for translation trading is related in the semantic description with Compatibility Characteristic, for which

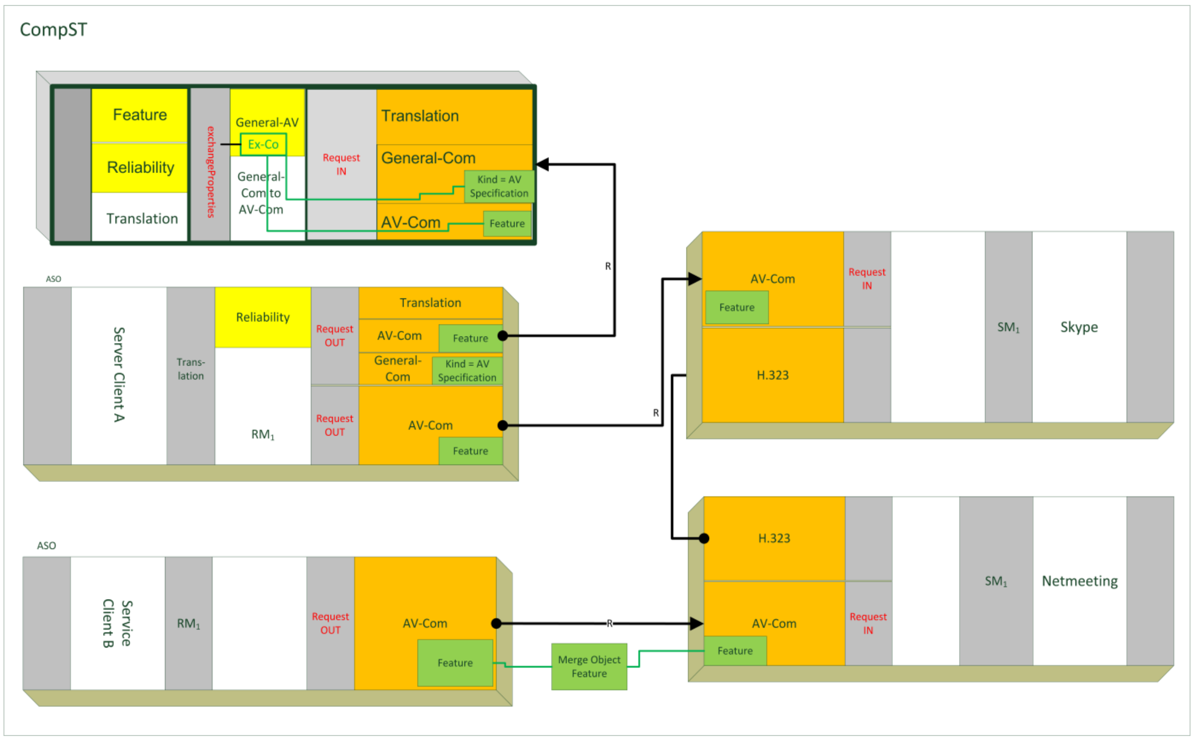


Fig. 48 - Audio-Video-Communication with alternative Translation

Translation Offers exist, then the Trader Agent will react automatically on the existence of these Compatibility Characteristics inside of a Trading Request (TRe).An alternative way of forcing early translation is shown in Fig. 47 - Alternative Example of Translation Offer.

The Request Port contains an additional Request Characteristic: “Translation”. This Request Characteristic demands to further Compatibility Characteristics in the Service Port, namely the two ones, which shall be mediated. A relationship established through the “exchangeProperties” Option-Slot is again illustrated. Obviously only a Translation Offer offering this service in one of its Service Modes will be discovered. Advantage is the avoidance of backtracking. However, the Composition Agent must already decide at the time point of initialisation of the ASO (cf. section 13.3) that a Translation Offer is needed. The CoA must adapt the set of references of Request Characteristics in the Request Mode, in order to include the Request Characteristic “Translation”.

A possibly resulting Composite Structure is shown in Fig. 48 - Audio-Video-Communication with alternative Translation. It is straight forward and can be easily understand in comparison to the original Service Composition in Fig. 46 - Audio-Video-Communication with Translation. However, something new happened here. Previously, the application of (some) Exchange Constraints had to be earlier, in order to have mediated information for the next Service Discovery (Trading Request) step. With the alternative translation method the even the processing order of the Service Ports becomes determined. The Service Port with the Request Characteristic “Translation” has to be performed first, in order to transfer the information to the Request Characteristic “AV-Com” in the second Request Port for the next Trading Request.

16 Case Study 2: Distribute Feature Composition (DFC)

Before the application of Semantic Characteristics of ACTAS for the Service Description (S-Model) and the Service Request (R-Model) is possible, ACTAS Administrators of diverse application domains have to become active and must introduce applicable characteristics. In this chapter, a simple example of Distributed Feature Composition in the domain of Technical Services for the telecommunication (cf. Example 21) is used for the development of Semantic Characteristics and their usage as "building blocks" for the Service Description and Service Request. The Example was chosen, in order to include the use of the Common Part of a Service Request and non-directed Service Composition.

The Example 21 - Distributed Feature Composition (DFC), gives a hint that in research areas, which are ruled by proprietary solutions of industrial companies (in this case companies for the production of telephone switchboards), the agreement on commonly accepted standards is not easy and might not be even wished due to company policies and compliance. A common research based on non-existing standards is not easy. For instance, every company can introduce their own features and Feature Description Languages (FDL) in the case of DFC. Two different approaches of FDL are discussed in the example.

An ACTAS Administrator in the domain of telecommunication will recognise that it could be useful to clarify the constraints originated in DFC in the context of telephone exchange services as early as possible, when he looks at the plurality of features and FDLs. However, he has to distinguish between properties, which are of interest for the customer, and other ones, which are more of use for the internal Service Compositions, which shall stay transparent for the Service Clients. In short, similar to the distinction between B2C and B2B for the e-business domain, he has to describe properties for the customer interfaces and other ones for the technical interfaces. ACTAS supports this distinction through the use of Request Characteristics.

The FDL offered by the site [Dee2011] clarifies the message sequences between the Technical Service of a telephone and the exchange. This is certainly a property, which is out of interest for the customer. However, if the exchange cannot support the wished features, there is no sense to go ahead with the establishment of the telecommunication connection. The second FDL, introduced in the publication [DeuKli2002], is declarative and based on the Domain-Specific Language (DSL) research and in the publication of [ShiAda et al.2010], it is shown and tested a way how the interpretation of this FDL and other DSLs can be adaptively realised through an SOA approach. Thus, the ACTAS Administrator sees a possibility to realise implementation instances, which can deal with a property containing feature interactions described through a program written in this declarative FDL.

Finally, the ACTAS Administrator could decide to introduce a Request Characteristic and a General Characteristic and call them both simply "Feature" (cf. equation (22-15) in Table 34 - Request Characteristics (RCh) and equation (22-6) in Table 32 - General Characteristics (GCh)). The introduced Semantic Characteristics contain distinct Char Properties based on the different

FDLs. The Request Characteristic is for the description of compatibility between a Service Request posed on behalf of a telecommunication customer, and a Service Offer offered by a Service Provider of a telephone exchange. For this purpose, the ACTAS Administrator will associate the Request Characteristic through its Semantic Description (cf. equation (22-15)) with the telecommunication customers as a user group.

The Request Characteristic "Feature" wraps a Char Property named "Dependency" (cf. equation (22-15)), which is assumedly based on the declarative FDL of [DeuKli2002]. The Char Property Class ("declarativeFDL") associated with the Char Property at its declaration could point to an implementation using the interpretation environment for the declarative FDL described by [ShiAda et al.2010]. The Value Constraints for the Char Property called "Dependency" could use two methods of the Char Property Class for the initialisation of this Char Property. The first method (assumedly called "setFDLprogram") would describe the constraints coming from the allowed feature interactions in the context of a given telephone exchange in a declarative description based on this language. A program for the composition of car parts is shown in Code 10 - Example for declarative FDL adapted from [KosMar et al.2008]. An initialisation with such a code is a typical initialisation of the Service Provider. The second method ("setFeatures") should enumerate the wished or offered features themselves. Like all methods used of Property Classes, this latter method could accept in its parameters a URL link to a file/resource describing the input. If the suggested initialisation file describes actually the wished or offered features will certainly depend on the application of the Request Characteristic in a Service Request or a Service Description, respectively.

In the next paragraph, the case study comes back to the FDL, which describes the features through message sequences. In the made assumption, the ACTAS Administrator could decide that the earlier introduced General Characteristic "Feature" (cf. equation (22-6)) wraps a Char Property called "SequenceDiagram". He thinks of the FDL offered by the site [Dee2011] as an at least proprietarily accepted standard, in order to clarify the compatibility between the service of the telephone exchange and the service of the telephone facility in the context of supported features. Thus, he associates a Char Property Class "sequenceFDL" with the Char Property "SequenceDiagram" at the time point of its declaration (cf. equation (22-6)), in order to take advantage of algorithms handling this kind of FDL. However, since he does not believe that anybody is interested in describing a Service Composition on the level of abstraction of the message exchange between telephone and switchboard, he will not introduce a further Compatibility Characteristic, but a General Characteristic, which can work with the previously discussed Compatibility Characteristic (a Request Characteristic is a Compatibility Characteristic) "Feature", in order to have access to the "agreed" features for an Exchange Constraint kept in the "merged" view of the Merge Property Object of the Service Property Dependency (cf. equation (22-6) and Definition 17 - View on Merge Property Object).

The "agreement" on certain features is a result of the application of the Merge Constraint, which will be applied on the property "Dependency" in the Composition Process (cf. section

13.5 - Step 4: Checking of Merge Constraints). The ACTAS Administrator associated the Merge Property Class “declarativeFDL-me” with the Char Property “Dependency” at its declaration in the semantic context of the previously covered Request Characteristic “Feature” (cf. equation (22-15)). The semantic context for both Semantic Characteristics is given through their Semantic Description. Both are related with ontological concept of a domain “telecommunication” (cf. equations (22-6) and (22-15) as well as Definition 5 - Semantic Description (SemDescr) of Char). The Request Characteristic is additionally declared as being usable by a “phoneCustomer”, a concept used in the assumedly existing ontology for the classification of user groups “$Ontology_{user}$”.

The Service Designer will use for the Service Description the Semantic Characteristics designed and published in the ontological repository by the ACTAS Administrators. This means, he will firstly describe Service Templates like the one shown in Code 11 and Fig. 49. The “works-with” relationships in the ontological repository of the Semantic Characteristics will help him to understand the relationships between the Semantic Characteristics. In Example 15, the Compatibility Characteristics “Phone” and “Audio-Com” as well as the General Characteristic “Audio-Phone” were covered. In this chapter, an ACTAS Administrator, assumedly employed in a telecommunication company and responsible for the support of DFC, might decide to introduce two distinct Semantic Characteristics, in order to address both aspects: a Compatibility Characteristic and a General Characteristic both called “Feature”. The Compatibility Characteristics “Audio-Com” and “Feature” were also declared as Request Characteristics, in order to use them for the description of a compatibility interface to the Service Request.

A possible Service Description for a telephone exchange/switchboard based on the mentioned Semantic Characteristics is shown in Code 11 and Fig. 49 - Service Template for the Feature Composition example. In opposite to the Service Template shown in Example 9, the illustrated ST in Fig. 49 has only one Service Mode. This Service Mode SM_1 has two Service Ports: SP_1 and SP_2 (the Service Ports in Code 11 have two indices, in order to show that they belong to SM_1). SP_1 offers a non-directed interface for the connection to another switchboard. This is a B2B like interface, which stays transparent to the Service Client(s). In fact, the Service Designer could describe a Service Mode without this interface in the assumption, that two telephone switchboard will have a connection via the worldwide telephone network. Nevertheless, ACTAS allows clarifying some additional parameters for this connection. Such kind of “phone” interfaces become even more interesting, in order to clarify telephone conferencing or the necessary use of a gateway, when the Service Clients have communication facilities linked to different networks.

The second Service Port is more important for the discovery of the Service Offer. It is declared as a Request Port through the “request” Option-Slot and as Server Port through the “direction” Option-Slot with the direction “IN”. The ST-Env_4 is shown with a Value Constraint for the initialisation of the Service Property “Dependency”, which was declared as Char Property in the semantic context of the Request Characteristic “Feature” as earlier described.

Example 21 **Distributed Feature Composition**

- Authorization Code (Authcode)
- AutoDial
- Auto Display
- Automatic Call Distribution (ACD)
- Busy Lamp Field
- Call Forward Busy
- Call Forward Don't Answer
- Call Forward Programmable*
- Call Hold*
- Call Hunt
- Call Park*
- Call Pickup
- Call Prompter
- Call Transfer*
- Call Waiting
- Consultation*
- Cutoff on Disconnect
- Executive Intercom
- Group Intercom
- Inbound Caller ID*
- Intercept Recording
- Last Number Redial
- Make Set Busy
- Message Waiting Indicator (MWI)*
- Multiple Appearance Directory Number (MADN) feature
- Music on Hold
- Name Display
- Outbound Caller ID
- Ring Again*
- Simultaneous Ring (SimRing)
- Six-Way Conference
- Speed Calling
- Symposium Call Center
- Three-Way Conference*
- Voice Mail
- Voice Mail: Announce Only

* Standard features

Table 23 - Telephone Features

Communication Services using telephone switchboards (also called exchanges) generate a feature composition problem, which is research subject of **Distributed Feature Composition** (DFC). DFC is a virtual architecture for specification and implementation of telecommunication services (cf. [JacZav1998]). In DFC, a **feature** is an increment of functionality, usually with a coherent purpose. In Table 23 several examples of features offered by an American telephone company are listed. Features are popular in various domains because they are easy to add and change. However, feature composition generates also the problem of feature interaction. A **feature interaction** describes the way, in which a feature or features modify or influence each other. Thus, feature interactions are part of defining overall system behaviour. It is possible to distinguish between the feature interactions on one side of the telecommunication (Local Feature Composition) and the feature interactions arising between the features on distributed sides of the telecommunication (Distributed Feature Composition).

Languages like for instance Feature Description Language (FDL) were developed, for the analysis of feature compositions. However, different FDLs with various purposes have been developed. Two approaches are discussed in this example. At the site [Dee2011], a FDL is offered, which allows the introduction of features for the description of message sequences between the phone and the exchange. In the publication of Deursen and Klint [DeuKli2002], a declarative approach of FDL is introduced in the context of Domain-Specific Languages. Their FDL is a declarative language that textually describes feature diagrams.

```
1.   module : customer, exchange
2.   processor : phone in customer
3.   processor : frontend in exchange, core in exchange
4.   feature "Call Setup"
5.        offhook : phone -> frontend
6.        dialtone : frontend -> phone
7.        digits : phone -> frontend
8.        setup_call : frontend -> core
9.        setup_complete : core -> frontend
10.       ringback : core -> phone
11.  endfeature
```

Code 9 - Feature Description and Message Exchange with FDL offered by [Dee2011]

The program in Code 9 is a simple example of the former FDL approach. It defines the exchange of messages between a “customer” and a telephone “exchange”, introduced through the “module” declaration in line 1. The “processor” statements in the next two lines define different entities within the customer and the exchange. The “customer” gets associated with a “phone” and the “exchange” shall contain a “frontend” and a “core” processor. These relationships are specified through the keyword “in”. The “feature-endfeature” block in line 4 declares the feature “Call Setup”. The feature block encloses all the interactions between the customer and the exchange, which are necessary for the realisation of the feature.

```
Car: all (carbody, Transmission, Engine,
     Horsepower, opt(pullsTrailer))
Transmission : one-of (automatic, manual)
Engine       : more-of (electric, gasoline)
Horsepower   : one-of (lowPower, mediumPower, highPower)
include pullsTrailer
pullsTrailer requires highPower

Result:
one-of (
     all (carbody, pullsTrailer, manual, highPower, gasoline, electric),
     all (carbody, pullsTrailer, manual, highPower, gasoline),
     all (carbody, pullsTrailer, manual, highPower, electric),
     all (carbody, pullsTrailer, automatic, highPower, gasoline, electric),
     all (carbody, pullsTrailer, automatic, highPower, gasoline),
     all (carbody, pullsTrailer, automatic, highPower, electric))
```

Code 10 - Example for declarative FDL adapted from [KosMar et al.2008]

The declarative FDL introduced in the publication of [DeuKli2002] is used for the adapted example of [KosMar et al.2008] in Code 10. In this code, the possible feature compositions for the assembling of a car are listed. In a publication of Shih-Hsi et al. of the California State University [ShiAda et al.2010], this declarative FDL can be interpreted as a Domain-Specific Language (DSL) through a SOA-based approach.

The two General Characteristics in the Service Mode part or the Common Part of the Service Description (i.e. “Audio-Phone” (cf. equation (22-3) in Table 32 - General Characteristics (GCh)) and “Feature” (cf. equation (22-6)) respectively) are examples for the use of Semantic Characteristics as “building blocks”. The use of the General Characteristic as “building block” was already discussed in 0. Its Exchange Constraint was adapted to the access of the Service

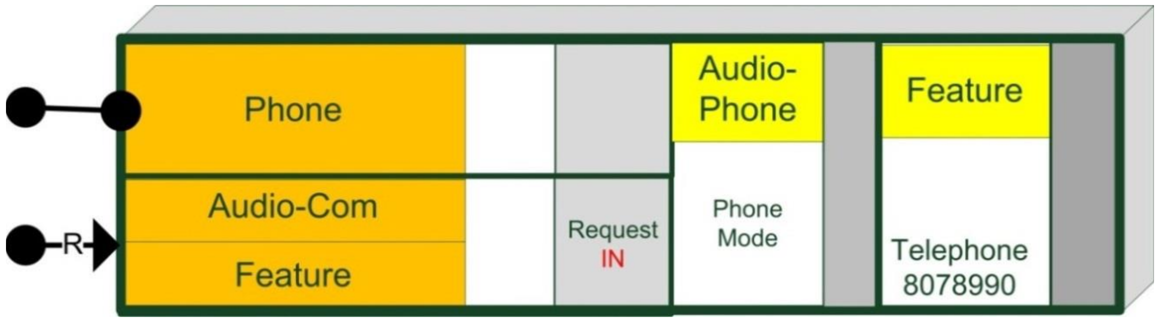


Fig. 49 - Service Template for the Feature Composition example

Properties of the Compatibility Characteristics "Phone" and "Audio-Com" with an "exchangeProperties" Option-Slot, which is again listed for the ST-Env$_2$.

In this Chapter, the General Characteristic "Feature" for the checking of the message sequence necessary for the realisation of the agreed features was discussed. The ACTAS Administrator provided even a Service Property "SequenceDiagram" declared in the context of this General Characteristic, in order to store the information about the necessary message sequences. An Exchange Property Class "sequenceFDLex" is used for the realisation of Exchange Constraint Char_Ex_Co$_1$ in the declaration of the GCh "Feature" (cf. equation (22-6)).

The "exchangeProperty" Option-Slot$_{2,1}$ in Code 11 adapts this Exchange Constraint (clearly referenced through $(\text{Feature}, Ex_{Co_1})$) and links the Service Property "Dependency" of the Request

$ST_{tele} = \left(ST^{\mathrm{ID}}_{\mathrm{tele}}, \mathrm{FA}^{\mathrm{Ref}}_{\mathrm{tele}}, \mathrm{SM}^{Set}_{tele}, \{Feature\}, \text{ST-Env}_1\right)$
$\mathrm{SM}_1 = \left(\mathrm{SM}^{\mathrm{ID}}_1, \mathrm{SP}^{\mathrm{Set}}_1, \{\text{Audio-Phone}\}, \text{ST-Env}_2\right)$
$\mathrm{SP}_{1,1} = \left((\mathrm{SM}_1, \mathrm{SP}_1)^{\mathrm{ID}}, \{\text{Phone}\}, \text{ST-Env}_3\right)$
$\mathrm{SP}_{1,2} = \left((\mathrm{SM}_1, \mathrm{SP}_2)^{\mathrm{ID}}, \{\text{Audio-Com, Feature}\}, \text{ST-Env}_4\right)$

$\text{ST-Env}_1 = \left(\text{Va-Co}^{\mathrm{Set}}_1, \text{Ex-Co}^{\mathrm{Set}}_1, \text{Option-Slot}^{\mathrm{Set}}_1\right)$
$\text{Option-Slot}_{1,1} = exchangeProperties\left((\text{Feature}, Ex_{Co_1}\right),$
$properties\left(\left[\left(\text{prop-ref}(SP_{1,2}, \text{Feature}, \text{Dependency}\right), \text{merged}, Features\right),$
$(\text{prop-ref}(ST_{tele}, Feature, SequenceDiagram), \_, Sequence)\right]\Big)$

$\text{ST-Env}_2 = \left(\text{Va-Co}^{\mathrm{Set}}_2, \text{Ex-Co}^{\mathrm{Set}}_2, \text{Option-Slot}^{\mathrm{Set}}_2\right)$
$\text{Option-Slot}_{2,1} = \text{exchangeProperties}\left((\text{Audio-Phone}, Ex_{Co_1}\right),$
$properties\left(\left[\left(\text{prop-ref}(SP_1, \text{Audio-Com}, \text{Audio-Quality}\right), \text{server}, AQ\right),$
$(\text{prop-ref}(SP_2, Phone, Speed), spView, S), \ (\text{prop-ref}(SP_2, Phone, Quality), spView, Q)\right]\Big)$

$\text{ST-Env}_4 = \left(\text{Va-Co}^{\mathrm{Set}}_4, \text{Ex-Co}^{\mathrm{Set}}_4, \text{Option-Slot}^{\mathrm{Set}}_4\right)$
$\text{Va-Co}_{4,1} = \text{va-co}\left(\text{prop-ref}(\text{Feature,Dependency}), [\text{setFDLprogram}([\langle\text{URL to file}\rangle])]\right)$
$\text{Option-Slot}_{4,1} = \text{direction}(\text{IN})$
$\text{Option-Slot}_{4,2} = \text{request}$

Code 11 - Service Description for telecommunication

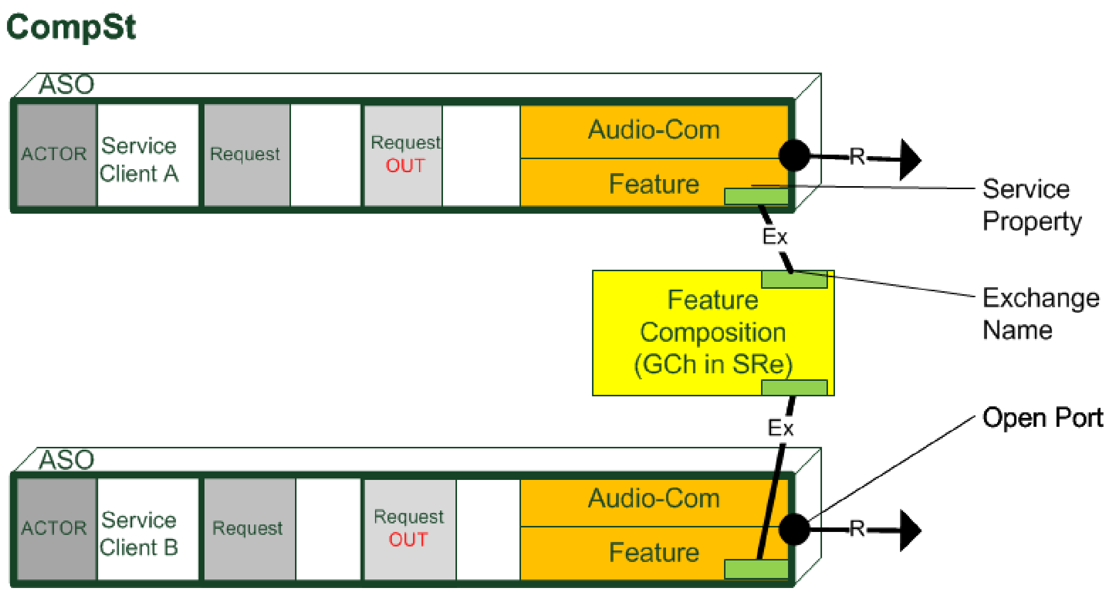


Fig. 50 - Initialised Composite Structure (CompSt) with global GCh from SRe

Characteristic "Feature" with the Exchange Name "Features" as well as the Service Property "SequenceDiagram" of the General Characteristic "Feature" with the Exchange Name "Sequence". (Please consider that the Property References (cf. Definition 15 - Reference of a Service Property) in this Option-Slot contain either the Service Port ($SP_{1,2}$) for the referencing of the Compatibility Characteristic or the Common Part (ST_{tele}) for the referencing of the General Characteristic, which have both the name "Feature".) In order to access only the "agreed" features, the link of the Service Property "Dependency" is done with the view "merged" (cf. Definition 17), i.e. the resulting information of the Merge Constraint in the Composition Process (C-Model) as previously said. The link between the Service Property "SequenceDiagram" of the General Characteristic "Feature" with the Exchange Name "Sequence" does not contain a view entry, since a Merge Constraint is only applied on the Service Properties of Compatibility Characteristics, in order to have an additional proof for the Service Compatibility.

Distributed Feature Composition (cf. Example 21) is a straight forward example for the use of the Common Part of the Service Request (cf. Code 12 - Service Request for DFC). The Common Part of a Service Request (SRe-Common in Definition 19) can contain a set of General Characteristics and environment information (SRe-Env in Definition 19). Following again the "building blocks" concept discussed in the S-Model, the administrator of the application environment, responsible for the design of Service Requests, can use additional General Characteristics and environment descriptions in the Common Part of the SRe, in order to define constraints and general information valid for several Client Requests or greater parts of the Composite Structure (CompSt). The Common Part of the Service Request becomes the Common Part of the CompSt data structure (cf. Definition 19 and Definition 24). In Fig. 50 and other figures, a General Characteristic originated of SRe-Common is shown with an additional yellow rectangle. The visibility of Service Properties in the Composite Structure is discussed more in detail, in section 13.3.3.

$$\begin{aligned}
&\text{SRe}_{\text{DFC}} = \left(\text{SRe}_{\text{DFC}}^{\text{ID}}, \text{ReA}_{\text{DFC}}^{\text{Ref}}, \text{SRe-Common}_{\text{DFC}}, \text{Client-Request}^{\text{Set}}\right)\\
&\text{Client-Request}_1 = \left(\text{RM}_1^{\text{ID}}, \text{PA}_{\text{A}}^{\text{Ref}}, \text{AST}_{\text{A}}^{\text{Ref}}, \text{RP}_1^{\text{Set}}, \{\quad\}, \text{SRe-Env}_1\right)\\
&\text{Client-Request}_2 = \left(\text{RM}_2^{\text{ID}}, \text{PA}_{\text{B}}^{\text{Ref}}, \text{AST}_{\text{B}}^{\text{Ref}}, \text{RP}_2^{\text{Set}}, \{\quad\}, \text{SRe-Env}_2\right)\\
&\text{RP}_{1,1} = \left((\text{RM}_1, \text{RP}_1)^{\text{ID}}, \{\text{Audio-Com, Feature}\}, \text{SRe-Env}_3\right)\\
&\text{RP}_{2,1} = \left((\text{RM}_2, \text{RP}_1)^{\text{ID}}, \{\text{Audio-Com, Feature}\}, \text{SRe-Env}_4\right)\\
&\\
&\text{SRe-Common}_{\text{DFC}}\\
&= \Big([\text{Feature Composition}],\\
&\Big(\text{Va-Co}_{Common}^{Set}, \ \text{Ex-Co}_{Common}^{Set}, \Big[\text{exchangeProperties}\Big((\text{Feature Composition, Ex-Co}_1),\\
&properties\Big(\Big[\Big(\text{prop-ref}\big((\text{RM}_1, \text{RP}_1), \text{Feature, dependency}\big), \ \text{merged}, \ \text{FeatureOne}\Big),\\
&\Big(\text{prop-ref}\big((RM_2, RP_1), \ \text{Feature, dependency}\big), \text{merged}, \text{FeatureTwo}\Big)\Big]\Big)\Big)\Big]\Big)\Big)
\end{aligned}$$

Code 12 - Service Request for DFC

The Distributed Feature Composition cannot be directly faced, without the Common Part of the Service Request (SRe-Common in Definition 19), because the Service Property "Dependency" in the Request Characteristic "Feature" can only check the Local Feature Composition. The information of several Service Properties "Dependency", containing the collections of features wished by diverse Service Clients of a requested Communication Service, has to be checked for Distributed Feature Composition. ACTAS allow the definition of Exchange Constraints in this direction in the Common Part of the Service Request.

In Code 12, the Service Request uses a General Characteristic "Feature Composition" (cf. equation (22-7) in Table 32 - General Characteristics (GCh)), which is settled like the previously discussed Semantic Characteristics in the domain telecommuniction. This General Characteristic supplies an Exchange Constraint, which will compare two feature collections, accessed through the Exchange Names "FeatureOne" and "FeatureTwo", through the method "checkDFC" of the assumed Exchange Property Class "declarativeFDL-ex". The method will check if the collections satisfy the conditions of the DFC. The parameters of this method could adapt the checking of the DFC to domain-specific settings similar to the initialisation of the Service Property "Dependency" for the checking of the Local Feature Composition through the Value Constraint $\text{Va-Co}_{4,1}$ in Code 11.

Again an "exchangeProperties" Option-Slot is used in Code 12, in order to link the Exchange Names of the "imported" Exchange Constraint with the Service Properties. These are just the Service Properties "Dependency" in the Request Characteristics of the two Client Requests. In section 13.3.3, is discussed future research, which allow the additional reference of Service Properties of the Composite Structure, which is still unknown at the time point of the generation of the Service Request by the application environment.

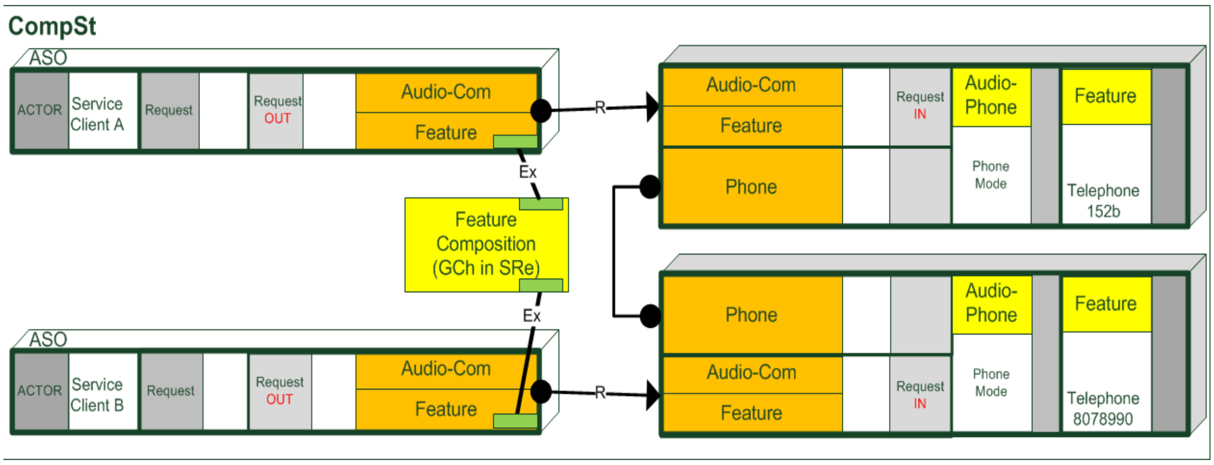


Fig. 51 - Composite Structure with global GCh for Ex-Constraint

Fig. 50 - Initialised Composite Structure (CompSt) with global GCh from SRe, shows how the information is used for the initialisation of CompSt, which is a data structure of the C-Model introduced in the next chapters.

Fig. 51 - Composite Structure with global GCh for Ex- – shows schematic a possible Service Composition of two Telephone Switchboards, which are linked through a common Phone standard (non-directed Service Composition with the Compatibility Characteristic “Phone”). The two Client Requests of the Service Request became the two ASOs. They were done on behalf of Service Client A and Service Client B. The combination of the two Request Characteristics “Audio-Com” and “Feature” in the Request Ports expresses that the (potential) Service Client are interested in an audio communication supporting the feature composition. The Local Feature Composition (cf. Example 21) can be tackled with Exchange Constraints defined in the Compatibility Characteristic “Feature”. The found services of Telephone Switchboards keep additionally a General Characteristic also called “Feature” in their Common Part. It is likely, that this General Characteristic contains further Exchange Constraints linked with the Service Properties of the Compatibility Characteristic “Feature”, in order to perform extended tests for the Local Feature Composition. Another General Characteristic “Audio-Phone” appears in the Service Mode part. It could contain just information about this Service Mode.

The figure shows nicely, that the Service Composition with the principal compatibility (cf. Fig. 51 - Composite Structure with global GCh for Ex-) leads to a selection of a Service Mode. (The example of a Service Template in Fig. 30 had two Service Modes for comparison. In the current version of ACTAS, a General Characteristic can only appear once.) Obviously, the Service Provider tackle the Local Feature Composition for all offered through writing the General Characteristic in the Common Part.

$\text{CompSt}_{\text{DFC}}=\left(\text{CompSt}^{\text{ID}}_{\text{DFC}}\text{ , Selected-SM}^{Set}\text{, ComProperty}^{Set}\text{, Merged-SP}^{\text{Set}}\text{, Object}^{Set}\right)$

$\text{Selected-SM}_1=\left(\text{RM}^{\text{ID}}_1\text{, SRe}^{\text{Ref}}_1\text{, ASO-RM}^{\text{Ref}}_1\text{, ModeProperty}^{\text{Set}}_1\text{, }\{\text{Mer-SP}_1\}\right)$

$\text{Selected-SM}_2=\left(\text{RM}^{\text{ID}}_2\text{, SRe}^{\text{Ref}}_2\text{, ASO-RM}^{\text{Ref}}_2\text{, ModeProperty}^{\text{Set}}_2\text{, }\{\text{Mer-SP}_2\}\right)$

$\text{Selected-SM}_3=\left(\text{SM}^{\text{ID}}_3\text{, SOER}^{\text{Ref}}_3\text{, SO-SM}^{\text{Ref}}_3\text{, Res-Info}_3\text{, ModeProperty}^{\text{Set}}_3\text{, }\{\text{Mer-SP}_1\text{, Mer-SP}_3\}\right)$

$\text{Selected-SM}_4=\left(\text{SM}^{\text{ID}}_4\text{, SOER}^{\text{Ref}}_4\text{, SO-SM}^{\text{Ref}}_4\text{, Res-Info}_4\text{, ModeProperty}^{\text{Set}}_4\text{, }\{\text{Mer-SP}_2\text{, Mer-SP}_3\}\right)$

$\text{Object}_1=(\text{SeqD\_Object}_1\text{, Char\_Property\_Object}_1)$
$\text{Object}_2=(\text{SeqD\_Object}_2\text{, Char\_Property\_Object}_2)$
$\text{Object}_3=(\text{Cl\_De\_Object}_1\text{, Char\_Property\_Object}_3)$
$\text{Object}_4=(\text{Se\_De\_Object}_1\text{, Char\_Property\_Object}_4)$
$\text{Object}_5=(\text{Me\_De\_Object}_1\text{, Merge\_Property\_Object}_1)$
$\text{Object}_6=(\text{Cl\_AQu\_Object}_1\text{, Char\_Property\_Object}_5)$
$\text{Object}_7=(\text{Se\_AQu\_Object}_1\text{, Char\_Property\_Object}_6)$
$\text{Object}_8=(\text{Me\_AQu\_Object}_1\text{, Merge\_Property\_Object}_2)$
$\text{Object}_9=(\text{Cl\_De\_Object}_2\text{, Char\_Property\_Object}_7)$
$\text{Object}_{10}=(\text{Se\_De\_Object}_2\text{, Char\_Property\_Object}_8)$
$\text{Object}_{11}=(\text{Me\_De\_Object}_2\text{, Merge\_Property\_Object}_3)$
$\text{Object}_{12}=(\text{Cl\_AQu\_Object}_2\text{, Char\_Property\_Object}_9)$
$\text{Object}_{13}=(\text{Se\_AQu\_Object}_2\text{, Char\_Property\_Object}_{10})$
$\text{Object}_{14}=(\text{Me\_AQu\_Object}_2\text{, Merge\_Property\_Object}_4)$
$\text{Object}_{15}=(\text{Cl\_Spe\_Object}_1\text{, Char\_Property\_Object}_{11})$
$\text{Object}_{16}=(\text{Se\_Spe\_Object}_1\text{, Char\_Property\_Object}_{12})$
$\text{Object}_{17}=(\text{Me\_Spe\_Object}_1\text{, Merge\_Property\_Object}_5)$
$\text{Object}_{18}=(\text{Cl\_Qu\_Object}_1\text{, Char\_Property\_Object}_{13})$
$\text{Object}_{19}=(\text{Se\_Qu\_Object}_1\text{, Char\_Property\_Object}_{14})$
$\text{Object}_{20}=(\text{Me\_Qu\_Object}_1\text{, Merge\_Property\_Object}_6)$

$\text{Merged-SP}_1=\left(\text{Mer-SP}^{\text{ID}}_1\text{, Cl-SP}^{\text{Ref}}_1\text{, Se-SP}^{\text{Ref}}_1\text{, true, MeProperty}^{\text{Set}}_1\right)$

$\text{Merged-SP}_2=\left(\text{Mer-SP}^{\text{ID}}_2\text{, Cl-SP}^{\text{Ref}}_2\text{, Se-SP}^{\text{Ref}}_2\text{, true, MeProperty}^{\text{Set}}_2\right)$

$\text{Merged-SP}_3=\left(\text{Mer-SP}^{\text{ID}}_3\text{, Cl-SP}^{\text{Ref}}_3\text{, Se-SP}^{\text{Ref}}_3\text{, false, MeProperty}^{\text{Set}}_3\right)$

$\text{ModeProperty}_{3,1}=(\text{Feature, SequenceDiagram, SeqD\_Object}_1)$
$\text{ModeProperty}_{4,1}=(\text{Feature, SequenceDiagram, SeqD\_Object}_2)$

$\text{MeProperty}_{1,1}=(\text{Feature, Dependency, Me\_De\_Object}_1\text{, Cl\_De\_Object}_1\text{, Se\_De\_Object}_1)$

$\text{MeProperty}_{1,2}=\begin{pmatrix}\text{Audio-Com, Audio-Quality, Me\_AQu\_Object}_1\text{, Cl\_AQu\_Object}_1\text{,}\\ \text{Se\_AQu\_Object}_1\end{pmatrix}$

$\text{MeProperty}_{2,1}=(\text{Feature, Dependency, Me\_De\_Object}_2\text{, Cl\_De\_Object}_2\text{, Se\_De\_Object}_2)$

$\text{MeProperty}_{2,2}=\begin{pmatrix}\text{Audio-Com, Audio-Quality, Me\_AQu\_Object}_2\text{, Cl\_AQu\_Object}_2\text{,}\\ \text{Se\_AQu\_Object}_2\end{pmatrix}$

$\text{MeProperty}_{3,1}=(\text{Phone, Speed, Me\_Spe\_Object}_1\text{, Cl\_Spe\_Object}_1\text{, Se\_Spe\_Object}_1)$
$\text{MeProperty}_{3,2}=(\text{Phone, Quality, Me\_Qu\_Object}_1\text{, Cl\_Qu\_Object}_1\text{, Se\_Qu\_Object}_1)$

Code 13 - CompSt for DFC

17 Case Study 3: Supply Chain, B2B Integration

In Motivation and Evolution of DIS middleware, Fig. 11, the Supply Chain scenario was discussed as a motivation of Enterprise Application Integration (EAI), the enhancement of DIS with business and process logic like Workflow Management Systems, in order to overcome the integration challenge. The following scenario was adapted from [FeKeZa2008]. RosettaNet[9] is an example for an EAI environment. It defines standardized partner interface processes (PIPs), which include standard intercompany choreographies (e.g., PIP3A4 Request Purchase Order), and the structure and semantics of business messages. Although such standards certainly enable B2B integration, they still suffer from several drawbacks. All partners must agree to use the same standards and often the rigid configuration of standards makes them difficult to adapt to local business needs.

This scenario is settled in the category of the third aspect of services introduced in the State-of-the-Art for the classification of services. The inherent complexity of services given through the Service Composition on process level, i.e. business processes as discussed in section 3.5, makes the B2B integration complicated. The B2B integration in supply chain environments and its challenges enforced the development of DIS, EAI, and SOC as illustrated previously in the thesis. In chapter 12 of [FeKeZa2008], the help of SESA (cf. section 5.2.3) in resolving interoperability problems between business partners is discussed with an example originated in the SWS Challenge[10]. Their example consists of several Service Providers offering various purchasing and shipment options for diverse products through an e-marketplace called Moon. An assumed Service Requester called Blue intends to buy and ship a specified product for the best possible price.

The short and simplified scenario of Example 22 shows the challenges; a Semantic Web Services Execution Framework like SESA has to deal with. Since these frameworks also support the later phases of the life cycle, they have to deal with the inherent complexity of services. In the scenario of [FeKeZa2008], they even avoided additional issues with the orchestration. They only considered Abstract Services as a direct category of Concrete Services. The issues become nearly impossible to handle, when for instance the orchestration is kept transparent or is even settled in the Execution Phase possibly due to company policies.

The settlement of non-functional properties is another issue. In principle, every description of an entity in WSML (the specification language of WSMO) allows the specification of non-functional properties. However, it is not closer specified what they are. Ontologies in WSML for the closer description of such non-functional properties are listed in the appendix. In the scenario of [FeKeZa2008], they considered user preferences with the help of non-functional properties.

[9] www.rosettanet.org

[10] www.sws-challenge.org

Example 22 **The SESA support of B2B integration**

Following services and applications are involved:

- Customer service is done through a Customer Relationship Management (CRM) system.
- An Order Management System (OMS) is integrated.
- The application of Blue follows the RosettaNet standard, which includes the standardization of choreographies like a Request Purchase Order, and business messages.

The integration of these systems through Web Services is given. WSMO engineers are assumed in the SESA scenario designing model services and requests and publish them in the Moon middleware repositories. They also define mappings between several ontologies published in the same repositories. However interoperability issues occur since engineers on the requester's and provider's side model services independently, meaning that they use different ontologies for the Capability Descriptions and diverse descriptions of the choreographies. Blue might do his design on the base of the Rosetta standard, whereas a Service Provider could use proprietary information and choreography specifications of his CRM/OMS system. This is just the problem stated in Problem Statement in chapter 6. An Autonomic SOC is hardly possible.

The solution developed with SESA, i.e. on the base of the WSMO standards in [FeKeZa2008] is innovative, but proprietary again in the end. Nevertheless it is an interesting solution, which should be used by Autonomic SOC, since the integration issues are certainly solved for the given environment. Therefore, the solution of SESA is shortly discussed in the following paragraphs.

Firstly, the business services are modelled as Web Services. The created Web Services serve as adapters to the mentioned environment (CRM/OMS and RosettaNet). They are described through WSDL files, including XML Schema for messages, definition of interfaces, operations, bindings, and end points. The adapters perform lifting and lowering functionality for XML Schema and ontologies needed for the Service Grounding (phase 5 of the life cycle of services) definitions of WSMO services.

The definition of the Service Grounding, i.e. the transformation of the semantic descriptions of WSMO to the syntactic descriptions of WSDL has to be done in a second step. This transformation was an integral part of OWL-S standard. In WSMO, Semantic Web Services Execution Frameworks like SESA offer a certain support. The results of the second step are Semantic Web Services and goals described according the WSMO definitions of these components (cf. section 4.4.3). The book has a closer look at this second step. It separates the description in the (1) creation of ontologies and grounding, (2) the creation of functional and non-functional descriptions, (3) the creation of interfaces and grounding, as well as the (4) creation of ontology mappings.

The several times stressed complexity of the dealing with the inherent complexity during this thesis becomes anew evident in the fact, that the scenario of SESA is restricted to the choreographic part of the WSMO interface description. The standards of WSMO distinguish between choreography for the Service Discovery and for the Service Execution phase. The former is also closer specified as "late-binding", when the Service Provider offers Abstract Services and the choreography has to be clarified with the Concrete Service, the instance of the service, in the later Service Grounding (phase 5 of the life cycle of the services). The execution choreography in the interface of a WSMO

> Service Description shall define the exchange of messages in the sixth phase of the life cycle.
>
> The scenario considers one case more closely, namely the so-called AchieveGoal execution. The Service Provider environment looks firstly for WSMO Service Descriptions stored in its repository, in order to discover Abstract Services matching the WSMO goal specifications of the received message from the user application. This can be seen as a kind of Service Trading (phase 2 of the life cycle of services). With the found Abstract Services the matching is extended to the known instances of services being in the category described through the Abstract Service (phase 3 of the life cycle). A Service Ranking and Selection is performed in the SESA scenario afterwards (phase 4 of life cycle). The late-binding choreography is used for the upsetting of the negotiation conversion between Service Provider and Service Requester in the Service Grounding phase. The Deployment Phase has to do a Process Mediation. A Data Mediation as added in the Execution Phase (phase 6 of the life cycle). The mapping rules for these mediations had to be stored in the middleware repositories doing the Service Discovery (inclusive the dealing with Abstract Services) in the Service Design phase (phase 1 of the life cycle), which was previously sketched.

Summarizing, the described solution of SESA left us with an approach, which works fine for the specific relationship, and it should continue to work fine until the involved parties decide to use new ontologies or choreographies. This decision for changing the outer parameters might not be arbitrary. A change of the standards like RosettaNet can lead to such a situation. Nevertheless, the organization of Blue could wish to extend its business relationships, on the one hand. It would prefer to use the given environment, but want to reach additional Service Providers. On the other hand, Service Providers, not knowing about the sketched solution, might be interested in offering their products/services. Here comes ACTAS into the game.

17.1 Offering the local solution in ACTAS

ACTAS closes the illustrated gap between the existing solutions and interested parties.For this purpose, it supports earlier phases of the life cycle of services, relies on given approaches, and allows the checking of several constraints, in order to have an early exclusion of non-matching services through its declarative environment. However, it only considers the Semantic Characteristics of the services and requests, which are seen as relevant for this early exclusion. On the next level of abstraction, ACTAS looks closer at the Service Properties declared in the semantic context of the Semantic Characteristics, in order to check the constraints with algorithms likely originated in the approaches themselves.

Let us get concrete in this scenario. The rumour of interested Service Providers and companies in the innovative solution of SESA may be heard by an ACTAS Administrator, at last. This administrator could exist in the company of Blue as well as on the side of the Service Provider. The first step, he has to do is the analysing of the solution through a classification by the four aspects of services.

The sketched solution obviously belongs in the domain of e-business. It describes the sale from the B2B perspective. This means first of all that points like extended warranty, which might be relevant in a B2C interface, are not considered. Secondly, a WSMO description can certainly not be handled by a normal costumer. The existence of an upper ontology $\mathtt{Ontology}_{\mathtt{Domain}}$ was assumed for this evaluation. This ontology might define a concept "ProductSaleB2B", which appears most appropriate for the domain description of the solution. The Service Design (1st phase, 4th aspect) is obviously done in WSMO. The use of the Abstract Service could mean the involvement of Service Trading (a criterion of phase 2 of the 4th aspect). The user preferences are integrated as a non-functional criterion (second aspect).

The ACTAS Administrator could find out several other criteria for the solution. A criterion for the third aspect could describe that different choreographies are used and adapted (e.g. RosettaNet or proprietary). Another criterion, settled in the fourth aspect fifth phase, could describe the use of the late-binding choreography or Abstract Services. A criterion, which also belongs to the fourth aspect, could specify that the solution is able to use the WSMO execution choreography in the Execution Phase. The latter criterion could be reflected in the existence of a Char Property that enables the checking of the matching of the execution choreographies. The diverse mapping of ontologies for the Process and Data Mediation could also lead to some criteria.

In the end, the ACTAS Administrator ends up with a bunch of criteria valid for the approach. The next step will be his pondering about the introduction of Semantic Characteristics and their Char Properties. For the declaration of the latter, he has to find out, which algorithms for their handling can be developed and offered. The introduction of distinct Semantic Characteristics and their Semantic Description (cf. Definition 4) is not independent of the available implementations. For instance, if he related a Semantic Characteristic in its Semantic Description with the handling of negotiation choreographies (such kind of negotiation was done in the scenario above) it would be appropriate to introduce a Char Property, which describes closer the used choreographies on both sides (e.g. the standard on which they are based). In this way, the Composition Process could check the supported choreographies through a Merge Constraint.

18 Case Study 4: Weather Forecast Scenario

In Example 23, a weather forecast scenario is described with several potential Service Providers and Service Clients settled in the domain of meteorology. It can easily be envisioned, how such kind of scenario can lead to a market for weather forecast or weather data providing services. Similar the Cloud Computing vision, advanced services and people could use these services. At last, (advanced) services could become Component Services for offering enhanced information and functionality of meteorology for Composite Service originated in various domains. The composition of interdisciplinary services would also have to observe a taking place of a dynamic change of Service Providers in these market places. As the previous case studies described the role of the ACTAS Administrator and Service Designer more closely, this case study is rather kept short, in order to carry over the idea of an adaptive, domain-general market place for advanced services to a similar scenario given in the fifth case study concerned with smart grids.

The offered services may describe the functional side of their services through different designs (e.g. WSDL, OWL-S, or WSMO). Through Compatibility Characteristics classified for the different Service Designs compatibility can early be ensured. A domain and application specific Compatibility Characteristic could be provided for the selection of the right Service Candidates. For an additional categorization of the world of services in the context of the weather forecast, Compatibility/Request Characteristics with non-functional Service Properties should be introduced, in order to check the reliability of the services or the compatibility of the output data of the supported weather models. The earlier discussed "Reliability" Request Characteristic could be applied in this direction. The location of the data providing services is important for the weather forecast services of the regions. Therefore, a Compatibility Characteristic classified for location recognition would also be of interest in the given context.

Example 23 **Weather Forecast Scenario**

> The existence of services for the calculation of weather forecasts for selected regions and a more global area is assumed. The calculation is based on weather models. The services need input data on wind speed, wind direction, air pressure, humidity, and many more. Further services could exist, which offer these weather data from multiple regions. The services for the calculation of the weather forecast shall rely on the data providing services.
>
> Each weather model is working with its own compilations of input data and integrates several algorithms adapted to various conditions, such as the climate, the current weather conditions, the location, and the time of the year. Thus, a given weather model may be appropriate for a certain region only at a defined period of time. The weather forecast for another region might work with another weather model. Changing weather conditions may make another weather model more appropriate for the calculation of a relevant weather forecast for a given region than other ones. Besides that the weather models have to be changed or adapted, the results of several fitting weather models have still to be weighted and chosen.
>
> Therefore, a service, calculating the weather forecast for a global area, is likely to include calculations of several services, which possibly offer only a forecasting for the weather of partial regions. The calculation of the weather forecast from data of partial regions demands an agreement on data compatibility with the output data of the regional weather models. Another potential goal of an advanced service could be a specialisation of the weather forecast like for instance a severe weather forecast. Finally, the Component Services for the calculation of advanced weather forecasts could be organised like a grid. The scenario can generally be interpreted as a world of services, i.e. a market place of services provided by different companies and offered on different levels of abstraction could be envisioned, which allows the permanent entry and exit of Service Providers.

In figure "Fig. 52 - Weather Forecast Scenario", a possible application of the MAS of ACTAS is scratched. Facility Agents offer pro-actively the services of the Service Providers. Basic services, providing for instance data of weather stations or satellites, could be accessed by various services operating with the weather models. Advanced services, in the figure one for severe weather forecast is portrayed, can rely on output data of these calculating services. ACTAS helps to discover and compose the (advanced) services. According to the introduction of the system environment, a Service Request starts in an application environment with its Request Agent.

The CoA will use Service Descriptions with a combination of the mentioned CChs/RChs, in order to exclude non-interesting service candidates as early as possible. Especially, the domain and application specific Compatibility Characteristics will make sure the discovery of Service Providers of the right service market. Nevertheless, the MAS of ACTAS can become active in an additional ways. The CoA could negotiate with the found FAs, in order to agree on Compatibility Characteristics, which are most appropriate for upcoming Service Discovery actions. In future research, results of the negotiation and its involved learning can be store in Actor Service Template (AST), in order to create more appropriate Actor Service Offers for the Service Clients/users. The Facility Agents of the Service Providers offering data for the weather forecast of specific regions could further allow an enhanced observation of the relevance of the used weather models. Depending on time of the year and the current weather in the regions some models might be more appropriate than others. Through the SOER an FA can “activate” or “deactivate” the according Service Modes of its Service Offers.

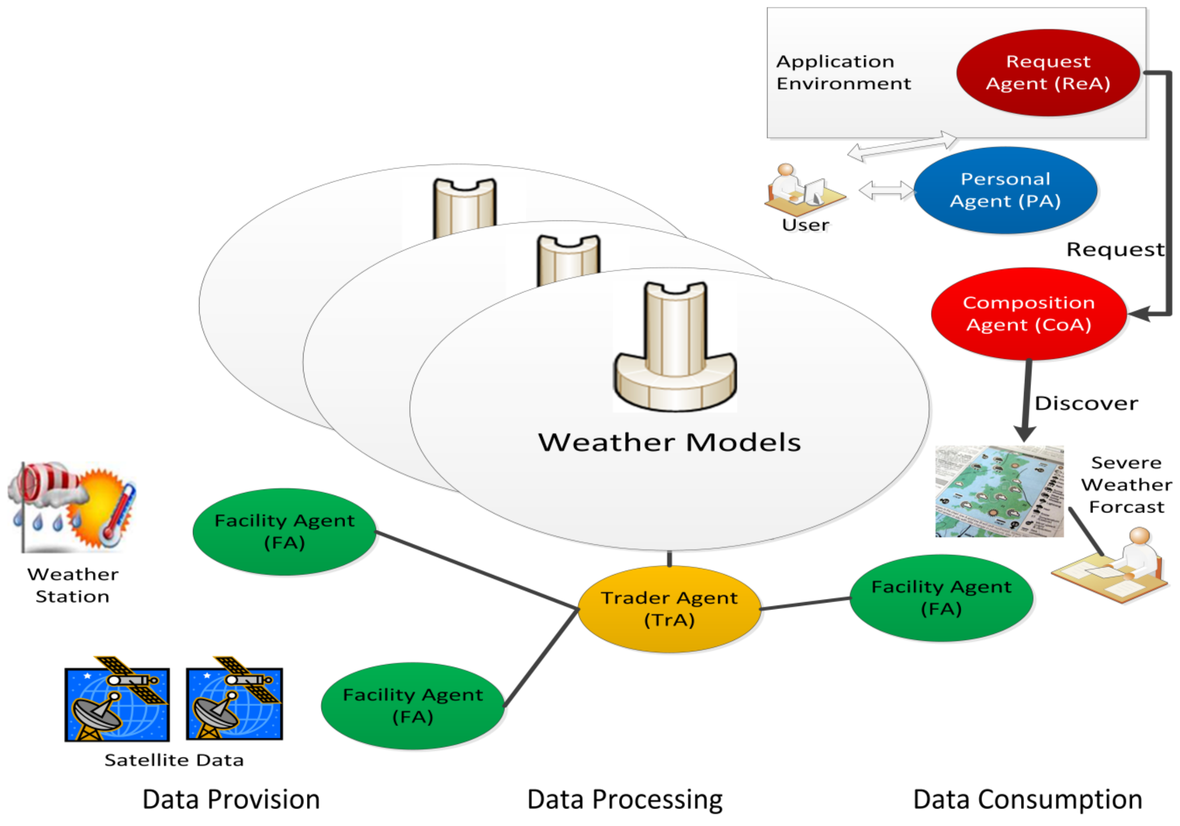


Fig. 52 - Weather Forecast Scenario

19 Case Study 5: Interpretation of Smart Grid as SOC

A **smart grid** is a generic term for digitally enabled transmission and distribution grids as used for power supply. The goal for the addressed grids is an autonomic adaptation towards the needed flow capacities and available resources as well as a changing set of participants (suppliers, network providers, and consumers). For this purpose including the guarantee of economics and reliability of the supply, data/communication networks became essential parts of a smart grid. A smart grid has to comply with the regulations and requirements of the involved markets as well as national and international laws. Proprietary solutions and policies of the companies establish further challenges for the autonomic design of smart grids.

In figure Fig. 53, a data network is shown as a central part of a smart grid. It functions as the digital enhancement of the technical network for the supply, an electrical grid in this case. Following the figure, the data/communication network could complement the technical components of the electrical grid with modules for the management of mobile and emergency workforces as well as metering. The data of these basic entities is used for the gathering and processing of information, as it is needed for smart metering or cost-processing. Global data networks allow the provision of advanced services like for example billing of the customers of various smart grids. Fig. 53 also coins the term "processing services". In this case study, a processing service is not necessarily visible to applications or Service Clients/Requesters, but it can be used as a Component Service of advanced services. For example, a cost-monitoring of a specific smart grid might be used as a processing service in an advanced service for the billing of the consumers.

In origin, the term smart grid was related with an electrical grid (cf. [AmiWol2005]). Due to an extended liberation of the power supply market and a similar interpretation of transmission and distribution grids, the term smart grid is increasingly applied to power grids in general. In this way, different energy carriers and systems (electricity, gas, and heat/refrigeration) can be linked with each other and be integrated into one comprehensive energy environment in a more efficient way. Besides the usage of storage power stations and rechargeable battery packs, the combination of gas and electrical grid allows additional ways for the necessary storing of the energy surpluses from non-controllable renewable energies like wind and solar. The smart grid idea might be extended to other grids like for example the ones used for the water supply, since they have to deal with similar challenges and they have their links to power grids as for instance, water gets heated, pumped, and used in storage power stations.

Development of flexible power grids has already commenced as it can be seen on various company sites (e.g. web interfaces of two power suppliers in UK and Germany[11]). *"The EU aims to fully integrate national energy markets by 2014 ... consumers can switch suppliers for gas and*

[11] http://www.nationalgrid.com/uk/LandandDevelopment/DDC/GasElectricNW/ or https://www.rwe.com/web/cms/en/183890/rwe/innovation/projects-technologies/power-and-gas-grids/

electricity, and suppliers must provide clear explanations of terms and conditions. Work still to be done includes aligning national market and network operation rules for gas and electricity as well as making cross-border investment in energy infrastructure easier." (Quotation from [Eur2013 p. 1]) It is estimated that double-digit billion euro investments will be needed in the coming years to expand the electricity and gas grids in the direction of the idea of smart grids.

Some key players and pilot projects of smart grids as well as smart homes in various countries are portrayed in [GunSah et al.2012]. The publication of Ardito et al. [ArdPro et al.2013] concentrates its view on the technical development of smart grids in Europe. Many of the research projects in the area of smart homes, especially smart metering, can be seen as supplementary for the research area of smart grids. Smart metering will be applied for the management of power consumption, in order to take advantage of low cost periods with a surplus of energy. The German concept of smart metering is different from the others. A multi-utility metering system is deployed: gas, water, and electricity meters are connected to a multi-utility communication (MUC) controller (cf. [GunSah et al.2012 p. 29]). With these controllers for a general power grid and the use of concentrators (cf. Fig. 53), the consumers can simultaneously use various power suppliers on the liberated market. However, this kind of smart metering demands advanced services for an efficient billing of the customers.

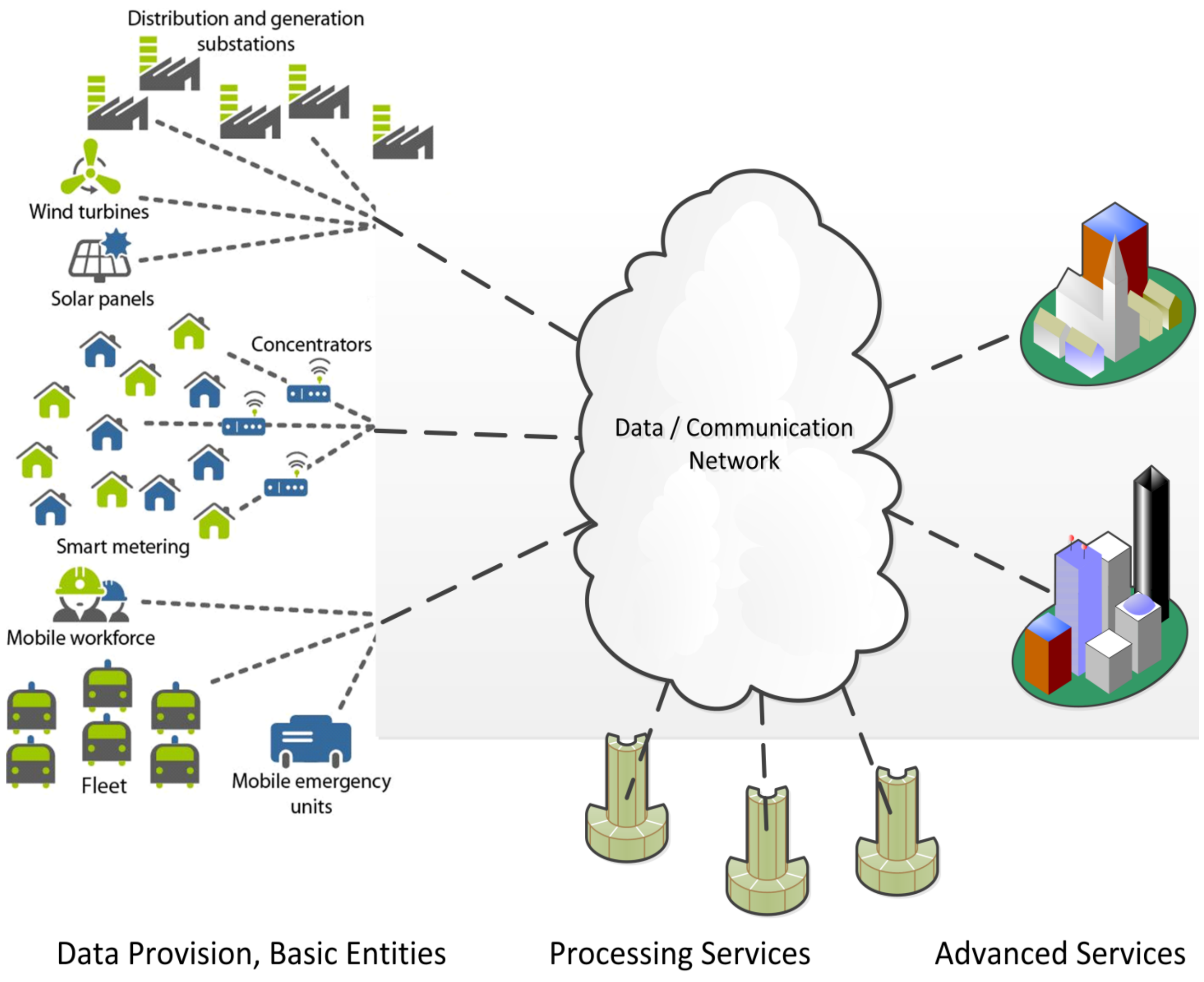


Fig. 53 - Smart Grid

The smart grid on a whole has to achieve a supply-demand match, which implies besides other reasons the low-latency communication and monitoring for some mission critical applications. An essential infrastructure like a smart grid also entails security issues. In the case of a smart grid, the security aspect does not only cover the usual data security coming with the communication network, but also integrates cyber and physical security of the electrical/power grid. According to [EroMou2013], this extended security aspect led even to a new supplementary research area of smart grids: smart grid forensic science. It is based on the experience that post-mortem analysis of a power system after a cyber-attack or natural disaster generally provides the most accurate comprehension of the causes. Its research goals are the protection against similar attacks in the future as well as the avoidance of failures during disasters.

19.1 SOC for a smart grid

It is a challenge to develop models for the smart grid research idea due to the different views on a smart grid. Following two approaches are mentioned as examples: (1) the **Open Smart Grid Protocol** (OSGP) [OSG2012], which is a family of specifications published by the European Telecommunications Standards Institute (ETSI), and (2) the **Smart Grid Architecture Model** (SGMA) [CEN2011] (cf. Fig. 54), which is a standardization proposal of joint European standardization organizations. OSGP is often used in conjunction with the ISO/IEC 14908 control networking standard for smart grid applications. In general, OSGP is an extension/adaptation of the Open Systems Interconnection Reference Model (ISO/OSI), which introduced standardized layers for the different levels of abstraction of networking/communication protocols starting from the application through security and transport down to the physical layer. With nearly 3 million OSGP compatible smart meters and other [smart grid] devices already installed in Europe, OSGP has become a [quasi] standard for smart meters and smart grid infrastructure communications in Europe according to [OSG2012 p. 11]. According to Fig. 54, the SGMA approach principally introduces new dimensions besides the technical ones originated in the OSI model. In a first additional dimension, the involved domains of a smart grid are addressed (Generation, Transmission, Distribution, Distributed Energy Resources (DER), and Customer Premises). In a second dimension, diverse “zones” of SGMA are presented for different levels of abstractions starting from the technical/process view and ending at the business/market view on a smart grid.

Increasingly, smart grids are interpreted as service-oriented environments. The publication of [LiaRod2013] proposes a service-oriented middleware for a partly implementation of a smart grid. In my publication [KlUnBr2012], I introduced a more general interpretation of the smart grid as a SOC environment, which involves a categorization of services through a semantic classification based on the aspects of services, particularly with regard to the semantic view on services in smart grids given with the discussed models. Thus, the fifth case study debates how ACTAS can improve the discovery of smart grid applications interpreted as service environments. The semantic classification related to the dimensions of the smart grid models achieves a more reliable compatibility description as well as the possibility to publish and compose services on different levels of abstractions. In its complexity, this case study extends the adaptive service market idea of the preceding fourth case study with its weather forecast related service environments.

Fitting to Fig. 54, ACTAS administrator could agree to introduce Semantic Characteristics standing for advanced services, which allow a data aggregation for the "market zone" as well as an advanced grid management on the level of the involved enterprises and the market. Other Semantic Characteristics could distinguish the processing services done by an adapted smart metering in the industry or at smart homes. The distinction between the supply network and the data network shall also be reflected in the classification of the Semantic Characteristics used in the smart grid scenarios of ACTAS. In fact, the shortly discussed dimensions of the SGMA model can lead to a useful, commonly agreed classification of the Semantic Characteristics. In this way, the Service Discovery can be done for services, which are clearly categorized through the semantics of a smart grid model like SGMA. For instance, an integration of technical devices into the power grid can be addressed in the SOC environment of ACTAS through Service Descriptions holding Semantic Characteristics classified as semantically belonging to the process level and the distribution sub-network. In Fig. 54, feeder automation and DER integration are mentioned in this direction.

In this case study, three examples are of closer interest: Example 24 - Wind Turbine Scenario, Example 25 - Consumer Scenario, and Example 26 - Billing Scenario. In Example 24, the technical network, an electrical grid built by possibly several Service Providers, has to integrate the service of a wind turbine, which is a technical entity that delivers energy in an unsteady way. The so-called consumer in Example 25 is the opposite entity to an energy delivering entity like the wind turbine. Thus, it could be modeled as a Service Requester/Client from this point of view. However, the role of the consumer in the technical network is more complex than that.

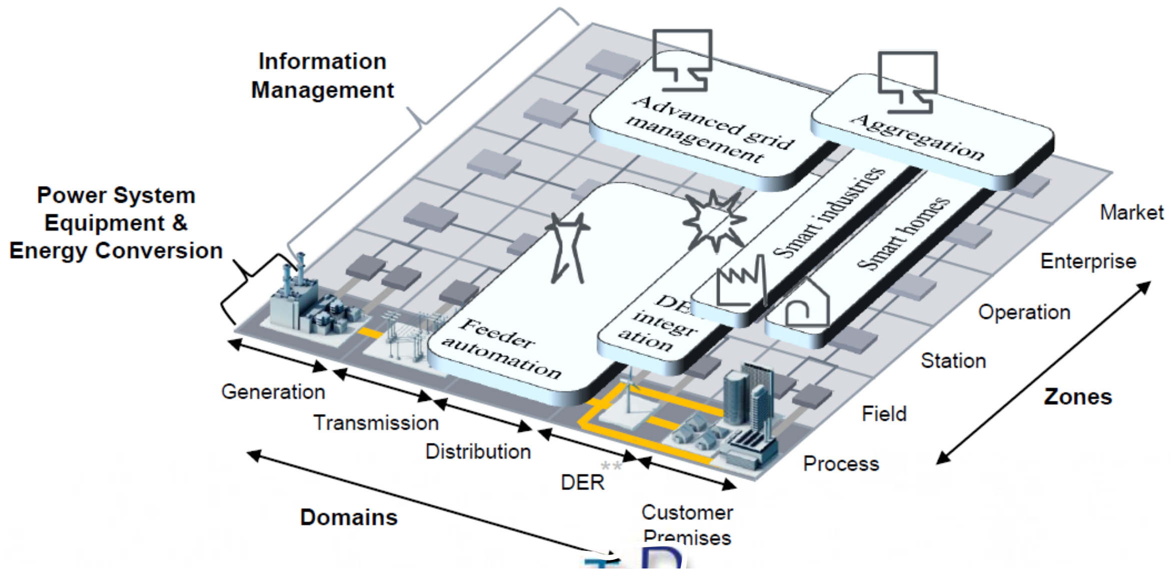


Fig. 54 - SGMA adapted from [CEN2011]

Therefore, its role gets reinterpreted, and the additional consideration of constraints is shown in this case study. Finally, the support of a discovery and composition of an advanced service is illustrated through Example 26.

Example 24 **Wind Turbine Scenario**

> A wind turbine and solar panels are examples of delivering sustainable, renewable energy. However, their power delivery depends on natural sources like wind or sun light. Thus, the electrical grid has to cope with the unsteady supply of energy, which can lead to periods of surplus or shortage of power in the grid. Additionally, the electrical grid can consist of several Service Providers in a liberated market. Thus, the delivery of energy might not only be split among distinct technical networks, but the environment might have to deal with different market interfaces built on the proprietary rules and laws coming with one specific Service Provider.

Example 25 **Consumer Scenario**

> A consumer for the supplied energy of a power grid is likely to have a contract with at least one Service Provider. Thus, he is bound by his contracts to certain power grids. Besides these commercial constraints, technical restrictions can occur. The customer can be connected to the distribution network in different ways and it might be compulsory that he is using fitting smart metering facilities.

Example 26 **Billing Scenario (Advanced Service)**

> In Fig. 54, aggregate services can be interpreted as advanced services in the sense introduced with Fig. 53. For instance, a corporate office, responsible for the service of customers of a liberated power market, will include in its application environment a billing service. The billing service might be offered as an advanced service by several Service Providers since it has to comply with different laws and a bunch of constraints. The billing service itself might aggregate data from several smart grids, to which a customer was connected in a given period. These data processing services themselves might rely on smart metering again.

The interpretation of a smart grid as based on SOC helps to deal with the challenges related with the liberation of the power supply market, where several smart grid Service Providers might compete. Each involved network needs its own expertise and possibly cannot be extended without planning, simulation, and testing. Therefore, the autonomic adaptation to a changing number of participants and resources on the one hand has to go together with the need of expertise for the various techniques and policies on the other hand. Considering these details of smart grids leads to a Service Provision that does not take place in a black box, i.e. the services have to be described on different levels of abstraction. A Service Request on a higher level of abstraction, which requests an advanced service providing a reliable and economic power supply, has to consider constraints on the service level of the smart grid: for instance the provider and consumer in Example 25 must be member of the same power grid, and the right data for smart metering must be provided through the data network. On one lower level of abstraction (in Fig. 54 the process level), the SOC of ACTAS could even describe the Service Composition of the technical services, e.g. the technically fitting components of an electrical grid or the gateways of different kinds of data networks. However, latest at this level of abstraction, the Service

Provider have to decide whether an adaptive discovery and composition of technical devices is wished and of advantage.

Alternatively, an enhanced service environment for smart grids can always take advantage of the expertise of existing solutions and reflects the availability for instance of technical services through offerings of services on a higher level of abstraction. In the end, it is up to the ACTAS administrator and the service designer of the Service provision to limit the service paradigm of the SOC environment of smart grids rather to the terminal/access points of the mentioned two networks without an extension to their technical services.

Nevertheless, services on a higher level of abstraction have their implications on the used networks, which should be reflected in the classification of their Semantic Characteristic s through the second aspect of services. Examples of such constraints for the needed features of the data networks are listed in Table 24 - Key requirements for the data network. An enhanced service environment should agree on the interpretation of these key requirements as non-functional parameters (nfp) for the Service Discovery. The mentioned quality of service (QoS) in this table is the standardized one for data networks as introduced in the section 2.3.2 - 2nd aspect of services: non-functional attributes. It is the ability to adapt data transfer priority and the according allocation of network resources to different applications, users, or data flows. A goal is to guarantee a certain level of performance to a data flow. In this case study, the term QoS is not restricted to data networks, but also used for the introduction of Semantic Characteristics for the categorization of advanced services.

Applications	Key requirements (2nd aspect of services)
Remote surveillance Remote control capabilities	High uplink throughput for remote cameras Real-time connectivity with high availability and low latency QoS Secure connections
Remote real-time monitoring	Real-time connectivity with high availability QoS Secure connections
Smart metering for residential and business locations	Support for a very large number of terminal devices QoS Secure connections

Table 24 - Key requirements for the data network

19.2 ACTAS for an enhanced service environment

For the support of pro-activity and autonomy, the service discovery framework ACTAS is based on software agents. Facility Agents (FA) and Trader Agents (TrA) are responsible for the publishing and trading of Service Offers as well as Service Templates. For instance, a member of the mobile workforce (cf. Fig. 53) has to deal with a changing access to terminals of eventually different kinds of data networks. Thus, even in times of smartphones, it might be a challenge for travelling workforce members to get connected to a data network fulfilling the key requirements as listed in Table 24. A software agent, in ACTAS preferable a TrA, responsible for the data network of the smart grid could keep track of the currently available possibilities of data transfer. The Personal Agent (PA) of a workforce member could act as a FA. Ideally, the current availability and capability of communication components is reflected in its Service Offers, whereas the Service Templates allow a planning with all potential components.

As an extension of this idea, each involved network (in Fig. 55 the supply network (Electrical Grid) and the data network) could be associated with a TrA, which handles the integration of services of technical components/facilities published by various FAs. The pro-active behavior of the Trader Agent must comply with the general features of the smart grid and the specific features of the managed technical network (e.g. national regulations and maximum flow capacity

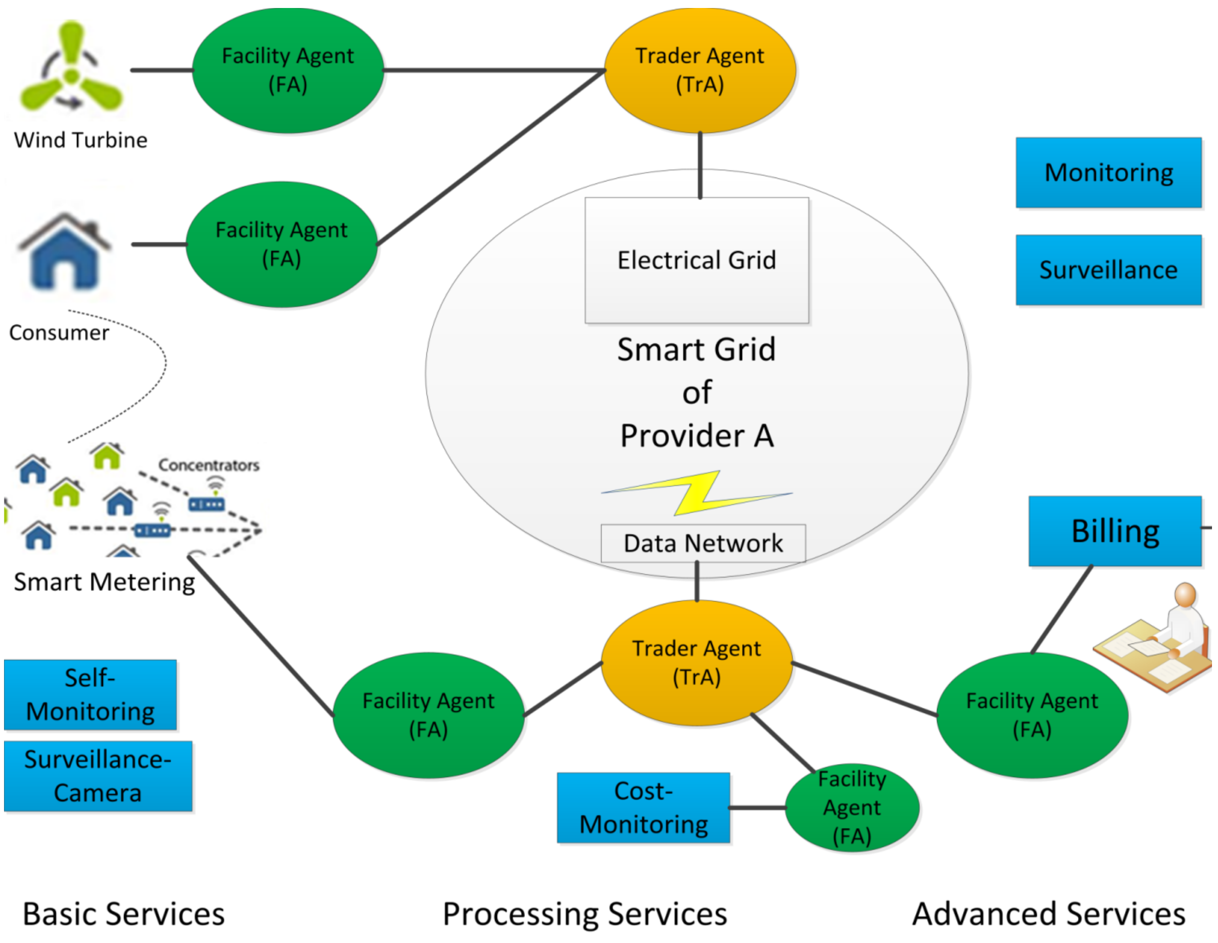


Fig. 55 - Smart Grid and ACTAS

must be observed). In accordance with its pro-active and re-active behavior, each TrA will discover the Service Offers of Candidate Services as they fit to the specific demands of its own network. In Fig. 55 - Smart Grid and ACTAS, the two shown Trader Agents belong to the Electrical Grid and data network of a smart grid provided by a Service Provider called A. Other Service Providers for smart grid might compete in a liberated power market. It is up to the trader to integrate the Candidate Services into the network. Alternatively, resources of not any longer needed services could be released. As in the examples introduced, constraints like a given contract, capability, or a given metering could be early checked.

Example 27 **Wind Turbine Scenario – continued**

A FA of a wind turbine could publish several Service Offers to the connected technical networks: one for the electrical transmission grid and other ones for its processing services, i.e. services like self-monitoring. The published Service Template would make the wind turbine known to the TrAs for planning. It is an advantage for the dealing with the changing availability of wind energy that the resources of the wind turbine are under control of just one software agent. Supporting the liberation of market, the wind turbine could be published through a Service Template with several Service Modes as a potential member of transmission grids of separate smart grids. The Service Offers reflect the current amount of produced electricity available to a certain Electrical Grid. A TrA might only "buy" a partial amount of this energy matching to the degree of capacity utilization. Additionally, the FA of the wind turbine could publish a functional Service Description via the data network offering a service for self-monitoring, i.e. the provision of monitoring information for its function control.

Example 28 **Consumer Scenario – continued**

Incorporating the TrA assumed to be responsible for the supply grid (Electrical Grid in Fig. 55); two alternative approaches may be discussed for its service interface to the consumers. On the one hand, the TrA could take over the role of a FA and offer the energy supply. On the other hand, the consumer themselves could be seen as services offering the function "to consume the energy". In the view of an adaptable, liberated market, the latter case might build a more homogenous model; since the facilities of the consumers get simply connected to terminals of the technical networks similar the ones of a wind turbine. The TrA might also check how the consumer is connected to its smart grid. In the case of a demanded smart metering, a proper data providing/processing service should exist. Assuming the constellation of Trader Agents as shown in Fig. 55, the TrA of the data network could act again as a FA offering smart metering services for the consumers, which are currently connected with the right facilities and features. In this way, the earlier introduced MUC controller and their concentrator function for the smart metering can be covered.

In Example 27, the wind turbine scenario is continued with the incorporation of the introduced Trader Agents of Fig. 55. In the case of offering a monitoring service (cf. Fig. 55), it might be an additional constraint that the TrA of the electrical grid integrated the wind turbine as a Component Service. However, such kind of an additional constraint is already covered through the Consumer Scenario (cf. Example 25), which is continued in Example 27. In the service-oriented view of the smart grid, the wind turbine service gets deployed and executed, in order to use the offered function through the data network for self-monitoring. However, in order to have a proper running service, the function has to be supported by the current features of the data network connected with the wind turbine (cf. key requirements in Table 24). Finally, the TrA of the data network itself could offer in a new role as a FA a processing service with accumulated self-monitoring data of several connected basic entities, which would be again usable for a control loop by the TrA of the electrical grid. Similar scenarios could be developed with surveillance or workforce services, which would not be directly related to a specific entity like a wind turbine but relevant for the smart grid at whole (cf. Fig. 53). Nevertheless, there might be constraints in particular smart grids that only wind turbines with existing surveillance cameras for security and existing workforce support get be integrated into a certain Electrical Grid. These additional constraints could be reflected in the Service Descriptions as discussed in the consumer scenario.

In Example 28, the consumer service is alternatively described from the same point of view like the service of the wind turbine in Example 27, i.e. it is offered by its FA and integrated into the Electrical Grid by the specific TrA. This is useful, since the main distinction between both services is simply that one is providing and the other one is consuming energy. Like the wind turbine the consumer should have contracts with the Service Providers of the potential supply grids, which can be used as a selective criterion through a Semantic Characteristic. The example also discusses the consideration of the constraint that the smart grid has to provide an additional service for the consumer with smart metering. Service Clients/Requesters can use advanced services for monitoring of the consumers' facilities or the billing, which might concern several smart grids, when the consumer rely on several kinds of service provision. An advanced service for billing is considered in the billing scenario, which is continued in Example 29.

Example 29 **Billing Scenario (Advanced Service) –continued**

> Through a FA, the corporate office might publish its billing service applicable for a group of consumers. A Service Request for such a kind of billing service (agreement on a Semantic Characteristic) will lead to a CoA that assumingly discovers the offered service. In this example, it is guessed that the service relies on processing services (called Cost-monitoring in Fig. 55) offered by the various smart grids, to which the consumers, given in the Service Request, are connected. The cost-monitoring of one certain smart grid might take advantage of the smart metering of its consumers.

The Fig. 55 - Smart Grid and ACTAS – lists as advanced services monitoring, surveillance, and billing. The request of advanced services initiates an application specific composition of currently available services, which will be done by a specific Composition Agent (CoA) in ACTAS. The monitoring and surveillance services might be used for several smart grids by applications for the management of power supply or security. Possible implications on the service provision of basic entities like the wind turbine and specific smart grids were shortly discussed earlier. The advanced service for billing is further discussed in Example 29. The applications, diverse services and Service Clients are connected to the data network in various technical ways, which have to fulfil key requirements as listed in Table 24. Therefore, the Service Description of the enhanced service environment should contain information about the currently supported or needed features. In the next section, a possible Service Description with ACTAS is portrayed.

19.3 Semantic Characteristics for smart grid scenarios

In the introduced three smart grid scenarios, ACTAS administrators will introduce Semantic Characteristics, which are semantically classified for (1) the advertisement of advanced/processing services, (2) the support of autonomic integration of basic entities into the involved networks, and (3) the guarantee of some non-functional parameters like the discussed key-features of the data network. Service Designers will use the Semantic Characteristics for Service Description, in order to categorize their services accordingly. Finally, the Service Provider will advertise Service Templates with potential Service Modes and adapting Service Offer Export Records through his FA(s). In this section, possible Semantic Characteristics for the smart grid scenarios are scratched.

In Table 25, possible Sematic Characteristics are listed with hints on their semantic classification. The kind of a Semantic Characteristic is revealed through the name affixes General Characteristic (GCh), Compatibility Characteristic (CCh), and Request Characteristic (RCh). Request Characteristics are a special kind of Compatibility Characteristics. According to the terms introduced in this case study, advanced services are advertised through Request Characteristics for possibly closer specified Service Clients, whereas processing services will be advertised through a simple Compatibility Characteristic, in order to be discovered as a Component Service (cf. section 9.1 - S-Model: Semantic Characteristics).

Examples of Compatibility Characteristics for the advertisement of processing services are $\text{CostMonitoring}_{\text{CCh}}$, $\text{SmartMetering}_{\text{CCh}}$, $\text{SurveillanceCamera}_{\text{CCh}}$, and $\text{SelfMonitoring}_{\text{CCh}}$ as listed in Table 25. In the last section, a general debate took place to the processing services of self-monitoring and the dealing with surveillance cameras as basic entities. The processing services for cost-monitoring and smart metering are used in the billing scenario (cf. Example 29). The classification of the Compatibility Characteristics shall ensure the right domain and the view on the service (1st aspect of services). At least the use in the context of a smart grid should be ensured. Since the mentioned Compatibility Characteristics advertise the processing services quite directly, a classification through the phase 1 of the 4th aspect, the service design, is a good idea, because an agreement on the used kind of Service Description of the actual services (e.g. OWL-S, WSMO, or WSDL) will improve the (principal) compatibility (cf. Definition 11 - Principal Compatibility for services and Service Ports).

At this point, it has again to be stressed that the Service Description of the service discovery framework ACTAS based on Semantic Characteristics is in fact an abstraction of Service Descriptions and criteria of actual services of service environments relayed through the FAs by the Service Providers. Although a classification of the Compatibility Characteristics for the Service Design might exist, it does not necessarily mean that the Service Description of an actual service is given in a Char Property declared in the Compatibility Characteristic. When such a

Char Property is declared, then Value Constraints will exist for the initialization with an appropriate Service Description (preferable located with a URI like in WSMO and propagated by Mr. Lee). It may be remarked that depending on the use of the Compatibility Characteristic, the Char Property might also be initialized with a Service Request, i.e. a goal in the case of WSMO. A Merge Constraint for this Char Property, which has to be declared in a Compatibility Characteristic (cf. Definition 4 - (Semantic) Characteristic (Char)), will later access an established algorithm (again preferable through a URI), in order to check the matching of Service Descriptions/Request (cf. section 10.4.2 - Merge Constraints).

The Compatibility Characteristics for the processing services are shown with further assumed Char Properties in Table 25. The Char Property for "Consumers" is imagined as holding the information of the consumers, who are involved with the particular service. The smart metering has further to clarify the current "grids" and the "contract". The cost-monitoring has to achieve an agreement on the "period".

In Table 25 - Semantic Characteristics, the Compatibility Characteristic for the cost-monitoring assumingly works with a General Characteristic for cost-monitoring (cf. works-with relationship in Fig. 28 - Principal ontological categorization of Semantic Characteristics, and in section 14.2 - Service Design - "Building Blocks" of ACTAS), which is the only General Characteristic closer discussed in this case study. In the billing scenario (cf. Example 29), the advanced service for the billing of consumers is supposed to take advantage of the processing services for cost-monitoring and eventually smart metering. The cost-monitoring was assumed to be done for a specific smart grid. Thus, possibly several cost-monitoring services have to be coordinated for the billing service. The General Characteristic and its Exchange Constraints, as discussed in preceding case studies, can support this coordination on the early state of Service Discovery done by ACTAS. For this purpose, it contains in our example the same Char Properties like the Compatibility Characteristic.

The advanced services for the billing of consumers can be categorized through the Request Characteristic $\text{Billing}_{\text{RCh}}$ (cf. Table 25). As presented in section 9.1, S-Model: Semantic Characteristics, the Request Characteristic is further classified with the user groups which contain the commonly agreed potential Service Clients/Requesters of the advanced services. For simplification, only one Char Property is mentioned: "Billing Data". It is assumed that Exchange Constraints will translate from this Char Property to the Char Properties of the cost-monitoring.

A Semantic Characteristic, used for the service-oriented modeling of the supply network, should contain specifications about the concrete smart grid like its identification, the kind of grid (e.g. gas grid, electrical grid), and possibly its location for the planning of the infrastructure. In our scenario, these specifications are kept in the Char Properties of the characteristic $\text{SmartGrid}_{\text{CCh}}$. It is worthwhile to mention that this Semantic Characteristic is also classified through the trading phase (phase 2) of the fourth aspect. As discussed in section 14.2.2 - Extended usability of Semantic Characteristics, this classification goes beyond a normal one, because it can also be used as an influence on the trading process. For instance, Trader Agents might react in a special way on services and requests, which hold a Semantic Characteristics classified through the phase 2 of the 4th aspect of services. Therefore, Service Offers, advertised with $\text{SmartGrid}_{\text{CCh}}$, are of interest for the introduced Trader Agents of the smart grid. The Semantic Characteristic $\text{SmartGrid}_{\text{CCh}}$ could have Char Properties holding information about the national and proprietary regulations of the smart grid (3rd aspect of services in the classification). Additionally, a specific Semantic Characteristic for a supply market of a certain country could be introduced (in Table 25 the Compatibility Characteristic $\text{SmartGridGe}_{\text{CCh}}$ for a German market is stated).

The entities of the supply network get their own Semantic Characteristics for a closer description and classification. In this case study, the Compatibility Characteristics for the wind turbine and for the consumer are introduced: $\text{WindTurbine}_{\text{CCh}}$ and $\text{Consumer}_{\text{CCh}}$. One of the suggested Char Properties covers the "Capacity", i.e. the amount of energy, which can be provided or shall be consumed respectively. Furthermore, information about the contracts with Service Providers of particular smart grids as well as the location of the entities could be supported for the selection of Service Candidates. The location is important for a wind turbine, in order to figure out, if the infrastructure of the supply network allows an efficient energy transport, since a smart grid in America will be not interested in an energy surplus of a wind turbine sited at the German coast line.

A basis of ACTAS is the categorization of the world of services through ontologically classified and commonly agreed Semantic Characteristics, in order to exclude as early as possible non-matching services from the set of prospective Service Candidates. Due to the abstraction of the published Semantic Characteristics used as building blocks of the Service Descriptions and Service Requests of ACTAS, the categorized world of services is not restricted to specific service environments; and Service Candidates of various SOC approaches can be pro-actively discovered and composed by software agents. In ACTAS, it is possible to look for services just fulfilling non-functional criteria, the 2nd aspect of services. In Table 25, two Semantic Characteristics simply named as QoS_1 and QoS_2 are introduced in this direction. They are not the standardized QoS criteria of data networks mentioned in Table 24, but semantically wrap agreed quality features usable in diverse contexts, which might be further restricted with classifications through the 1st aspect of services. The combination of several Semantic Characteristics results in the intersection of the sets of Service Candidates linked with each category of a single Semantic Characteristic given in the combination. Thus, QoS_1 can reduce the set of entities integrated into the supply network with the Semantic Characteristic $\text{SmartGrid}_{\text{CCh}}$ to the ones, which fulfill certain pricing criteria. QoS_2 might be useful in combination with the billing service, in order to guarantee some fiscal features.

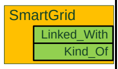
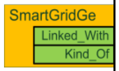
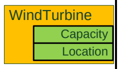
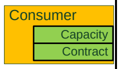
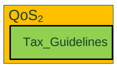
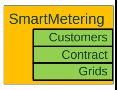
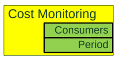
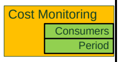
Semantic Characteristic	Classification	Char Properties
SmartGrid$_{CCh}$  (SmartGrid$_{GCh}$)  SmartGridGe$_{CCh}$	1st aspect: domain smart grid, supply network (3rd aspect: laws, policies, SmartGridGe$_{CCh}$ for German laws) 4th aspect, phase 2: TrA for smart grids	Identification of the smart grid and its technical network (Linked_With, Kind_Of)
WindTurbine$_{CCh}$ 	1st aspect: specific technical component of electrical grid	Capacity - Information includes its current free power capacity and its location Location and further information
Consumer$_{CCh}$ 	1st aspect: specification of electrical grid and the special consumer view	Capacity - Information about the wished/current power consumption Contract – Includes information about smart grid / Service Provider
QoS$_{CCh}$  QoS$_1$ Price_Period Quality	Mainly classification through 2nd aspect of service, i.e. nfp	Specifications for the technical service of the data network
SmartMetering$_{CCh}$  (SelfMonitoring$_{CCh}$, SurveillanceCamera$_{CCh}$)	1st and 2nd aspect: technical component of smart grid with self-monitoring, smart metering, or surveillance function, respectively 4th aspect, phase 1: the used functional description	Costumers – The consumers for whom the smart metering is given Contract – Conditions Grids – Smart Grids for which the smart metering is given (Eventually Service Description)
CostMonitoring$_{RCh}$,   CostMonitoring$_{GCh}$	1st aspect: interface of the smart grid for cost-monitoring 4th aspect, phase 1: the used functional description	Consumers – The consumers for whom the cost-monitoring is given Period for cost-monitoring (Eventually Service Description)
Billing$_{RCh}$ Billing Billing Data	1st aspect: user interface – billing for the consumers of smart grids 4th aspect, phase 1: the used functional description	Billing Data (Eventually Service Description)

Table 25 - Semantic Characteristics

19.4 Service Descriptions for smart grid scenarios

With the publication of the Semantic Characteristics, the Service Designers on the side of the Service Provision will have a tool box of building blocks for their Service Descriptions. They might decide to offer Service Templates with several Services Modes, in order to cover for example distinct contracts of a wind turbine with possibly diverse power suppliers separately. In this section, the three grid scenarios are continued with the discussion of Service Descriptions. Service Requests and the Service Composition are tackled in the subsequent section.

Example 30 **Wind Turbine Scenario – continued**

The Service Template in Fig. 56 has two Service Modes for the wind turbine. Both Service Modes have only one Service Port, which is declared as an IN Port through an Option-Slot. Obviously, the Service Designer intends to address in the second Service Mode the compliance with the German smart grid market. In the common part of the Service Description a General Characteristic for a closer description of the owner of the wind turbine is sketched. The Service Designer will include Value Constraints into the environments of the Service Description, in order to set accordingly the Char Properties, which are addressed as Service Properties in a Service Description. Additionally, Service Offer Export Records (SOER) are designed for the adaptation of the Service Template through the publishing FA. For instance the current capability of the wind turbine could be adapted.

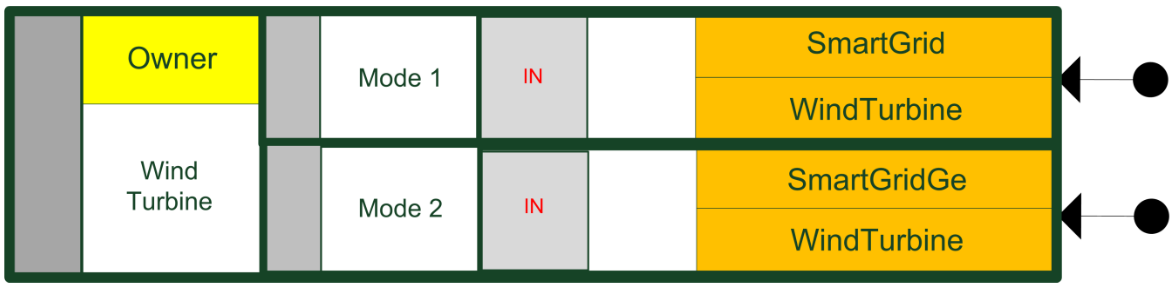


Fig. 56 - Service Template/Offer for wind turbine scenario

The Compatibility Characteristics SmartGrid$_{CCh}$ and SmartGridGe$_{CCh}$ were introduced and classified for being discovered by a TrA of a supply network. Through their classification with the 4th aspect, phase 2, they are quasi flagged for the interest of this kind of software agents. Thus, the TrA of the electrical grid will look for services advertised for a grid of its kind. In combination with a Semantic Characteristic giving a closer service description of a basic entity of the smart grid, the Service Port of a Service Template can be designed. Through the principal compatibility a TrA responsible for the supply network will be able to discover matching services of basic entities. Eventually, the Trader Agent will check more closely the Merge Constraints of the Char/Service Properties of the Semantic Characteristics of principally compatible Service Ports (cf. Fig. 60 - Principal Compatibility with applied Merge Constraints).

The Service Designers for the wind turbine or consumer scenarios could use fitting combinations in the Service Ports of their Service Descriptions. This is shown in Example 30 for the wind turbine and in Example 31 for the consumer. Example 31 demonstrates through an additional Service Port the consideration of a constraint demanding the existence of smart metering that fulfils given quality criteria. In the next section, the wind turbine and the consumer scenario will come to a conclusion with the Example 33, showing the support of ACTAS for the TrA of the electrical grid and the integration of the basic entities.

Example 31 **Consumer Scenario – continued**

Similar to the Service Description in Fig. 56, the Service Description of the "consumer service" in Fig. 57 has a Service Port, which is declared as an IN Port. It contains a combination of two Semantic Characteristics. One is the $\text{SmartGrid}_{\text{CCh}}$ and the other Compatibility Characteristic ($\text{Consumer}_{\text{CCh}}$) stands for a closer description of the consumer. However, in this scenario an additional constraint shall be observed. Therefore, the Service Mode contains a second Service Port declared as an OUT Port, which asks for a smart metering ($\text{SmartMetering}_{\text{CCh}}$) that fulfils the special quality of service features as specified by the value constraints in the Service Properties of the Semantic Characteristic $\text{QoS}_{1,\text{CCh}}$.

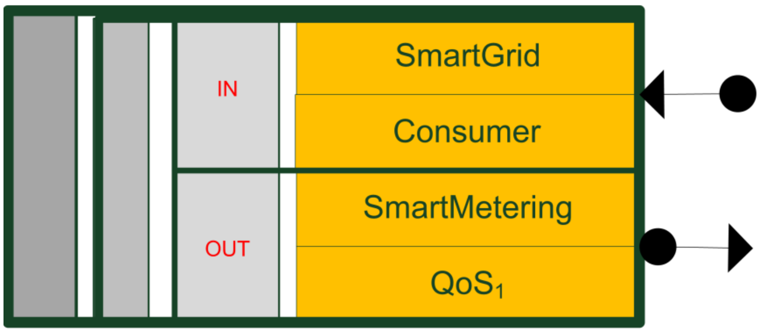


Fig. 57 - Sketched Service Mode of the "Consumer Service"

The continuation of the billing scenario in Example 32 shows a possible Service Template, which allows various Service Providers to offer a billing service as an advanced service visible for a Service Request generated in an application. The Service Template contains in its common part a commonly agreed description of the Service Provider through a General Characteristic (cf. Fig. 58) that might be helpful for the selecting of Service Candidates. The shown Service Template has two closer discussed Service Modes. In the next section, the initiation of the Service Request in an application and the Service Composition is of interest.

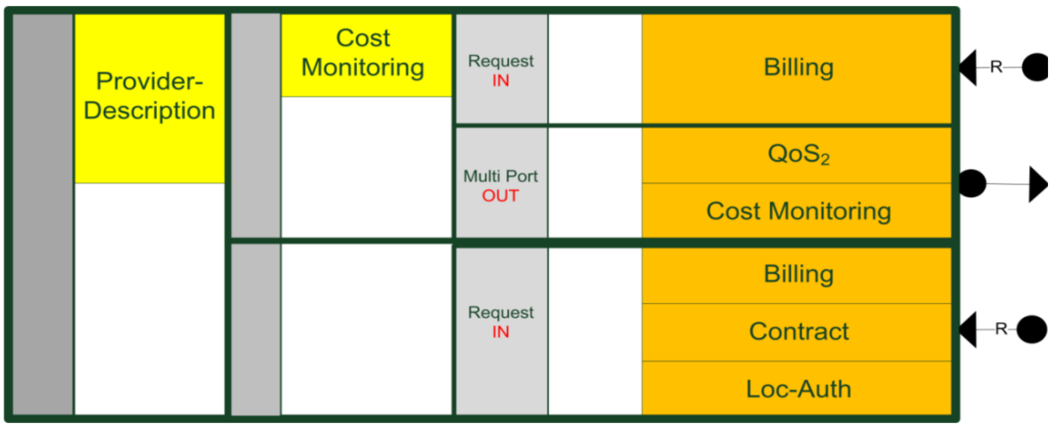


Fig. 58 - Service Template for billing service

Example 32 **Billing Scenario (Advanced Service) – continued**

As an advanced service, the billing service (Billing$_{RCh}$) is directly offered for Service Clients through Request Ports (Option-Slots “Request” and IN Port) given in the two Service Modes of the Service Template shown in Fig. 58. In the upper Service Mode of this figure, just the Request Characteristic for the billing service is demanded for the principal compatibility. However, through a second Service Port, which was designed as an OUT Port, the cost monitoring service with a specific quality of service is requested (CostMonitoring$_{CCh}$, QoS$_{2,CCh}$). As explained already in Example 29, the cost monitoring service is assumingly a processing service of a certain smart grid; and it can be that several cost monitoring services have to be discovered and composed, in order to do the billing for all enumerated consumers, especially when a consumer is actually a customer of several smart grids. Therefore, this OUT Port has additionally the Option-Slot setting for a “Multi-port”, i.e. the Service Port can be used for integration of several compatible Component Services. The General Characteristic CostMonitoring$_{GCh}$ (portrayed in yellow in the figure) as introduced in Table 25 - Semantic Characteristics, can be used with its Exchange Constraints for the “coordination” of the Service Properties of the Compatibility Characteristics “Cost Monitoring” appearing in the Service Ports of the found Composite Services. It is future research to improve the flexibility of the Exchange Constraints, in order to improve the dealing with a flexible number of Semantic Characteristics occurring in such scenarios.

The lower Service Mode illustrated in Fig. 58 has only one Request Port for the billing service. Nevertheless, this Service Port includes two further Request Characteristics, which are not closer classified in this case study, although one is called “Contract$_{RCh}$” like an earlier discussed Compatibility Characteristic (cf. section 19.3). Obviously, there shall be a given contract for the billing service as well as an authentification and location check (Request Characteristic “Loc-Auth$_{RCh}$”) for the (principal) compatibility.

19.5 Application of the Service Descriptions and Requests

The Trader Agent of the electrical grid can react on or look specifically for services offered with the Compatibility Characteristics SmartGrid$_{CCh}$ or SmartGridGe$_{CCh}$ as discussed in the preceding sections. It is up to the behaviour of the TrA to determine what happens with the found information. On the one hand, the TrA could just be interested in the Service Templates and the rest of the scenario takes place out of control of ACTAS. On the other hand, the TrA could take advantage of the Service Offers and the involved resource management of the publishing Facility Agents, in order to achieve an effective availability control. In this latter case during the composition process, the TrA can also check additional constraints as they are assumed in the consumer scenario. Therefore Example 33 shows a possible conclusion of the consumer scenario with a Composite Service illustrated in Fig. 59 - Consumer Scenario applied by TrA.

In the billing scenario, it is assumed that a corporate office is running an application, which generates a Service Request for the billing of varying groups of consumers of possibly diverse sets of smart grids. This means, that the Request Agent of this application will create a Composition Agent looking for the introduced billing service. A continuation and conclusion of the billing scenario in this way is shown in Example 34. Nevertheless, the possible Service Request illustrated in Fig. 61 has an alternative Request Mode (RM) or Client Request for the Service Client A looking directly for the cost monitoring service. Alternative Client Requests were

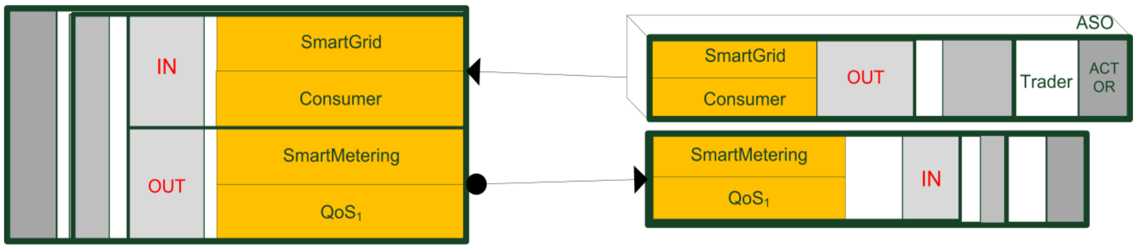


Fig. 59 - Consumer Scenario applied by TrA

discussed in Example 16 and section 13.2 - Step 1: Getting Information. It is the assumption of the billing scenario that the cost monitoring service is directly offered by a FA of a certain smart grid. A closer description of the requesting application might be given through a commonly agreed General Characteristic as shown in the common part of the Service Request in Fig. 61 ($\text{Application}_{\text{GCh}}$). In this case study, it was assumed that the application would be a corporate office for advanced consumer services. Further General Characteristics describing Exchange Constraints for the whole resulting Composite Structure can be given in this common part as discussed in section 13.3 - Step2: Initialisation of the Composite Structure.

Example 33 **Consumer Scenario – concluded**

In order to check an additional constraint occurring in the Service Description of a basic entity like the consumer in the consumer Scenario, the TrA can become an actor. Thus in Fig. 59 - Consumer Scenario applied by TrA, the Actor Service Offer (ASO) for the trader agent appears. In earlier discussions such an ASO appeared only for a Service Client of a Service Request. However, in this case study the trader agent uses pro-actively the resource management and availability control of the publishing Facility Agents. It will be future research, to clarify even further the role of an actor in ACTAS and to integrate learnt information with Actor Service Template (AST) (cf. section 8.6, Phase 6 – Service Execution and Feedback, and Definition 19). After the Service Composition of Fig. 59, the FA will adapt its Service Offers accordingly, since the consumer should receive an amount of energy of the electrical grid represented through the TrA. Additionally, it should be given that the consumer is connected through smart metering with the electrical grid. The Facility Agents of the consumer and the smart metering service of the smart grid can further support the deployment and operating of the smart metering and energy delivery.

In this case study, only the principal Service Composition of the billing service is further described. Nevertheless, the figure Fig. 60 - Principal Compatibility with applied Merge Constraints sketches the consideration of Merge Constraints for the Service Properties similar the figure Fig. 32 - Me-Constraints for directed and non-directed composition in section 10.4.2, which discussed more closely the application of Merge Constraints. As the smart grid scenarios could also include technical services for the supply networks, the Fig. 60 shows similar to Fig. 32 a non-directed Service Composition, too. In a supply network for water or gas, ACTAS could help with the planning of pipeline usage considering the specific connection, their capacity, and other qualities.

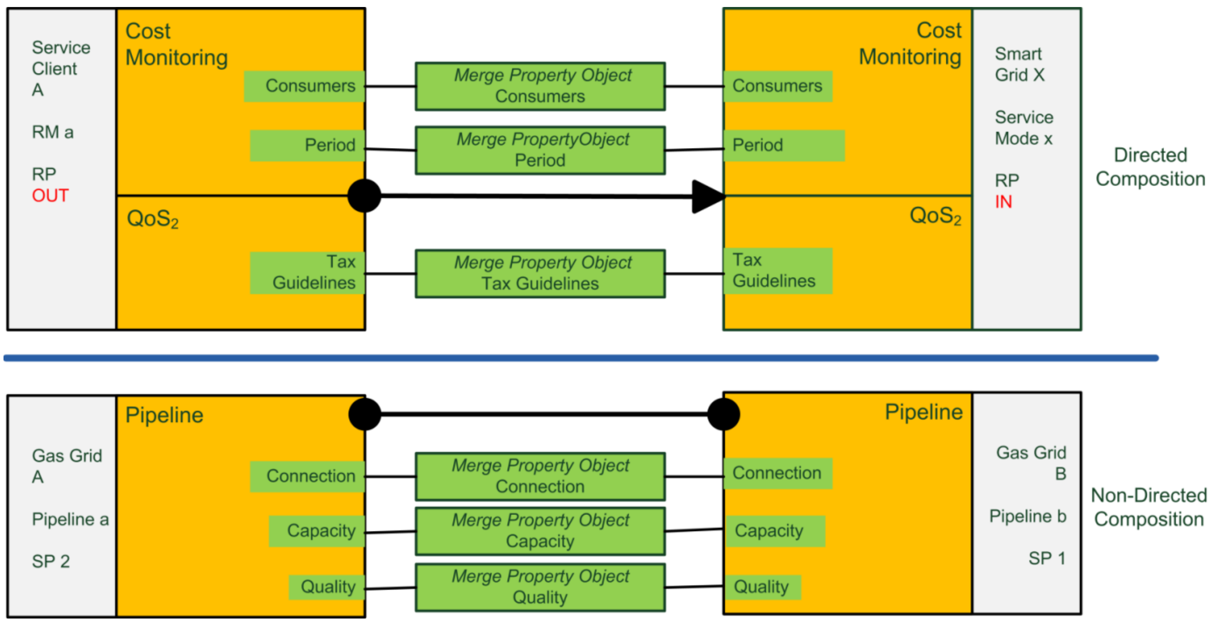


Fig. 60 - Principal Compatibility with applied Merge Constraints

Example 34 **Billing Scenario (Advanced Service) – concluded**

The shown Service Request in Fig. 61 has two alternative Request Modes, Client Request, for a Service Client A. Both have one Request Port with the Option-Slots for “Request” and OUT Port. In fact, there are no Request Modes or Client Requests for other Service Clients in this Service Request. The first Request Mode is looking for the billing service through a Request Port that just contains the Request Characteristic ($\text{Billing}_{\text{RCh}}$) from Table 25 - Semantic Characteristics. The other Request Mode has in its Request Port a combination of two Request Characteristics ($\text{CostMonitioring}_{\text{RCh}}$ and $\text{QoS}_{2,\text{RCh}}$). In Table 25, these Semantic Characteristics are only listed as Compatibility Characteristic, since the scenarios of this case study assumed the cost monitoring service rather as a processing service of a specific smart grid. It was not directly planned to offer this service directly to a Service Client through a Request Characteristic. In this sense, the illustrated Service Request is an extension of the scenario. It is hinted that a General Characteristic in the Request Mode might help out in providing fitting Exchange Constraints for the dealing with the constraints of the cost monitoring service.

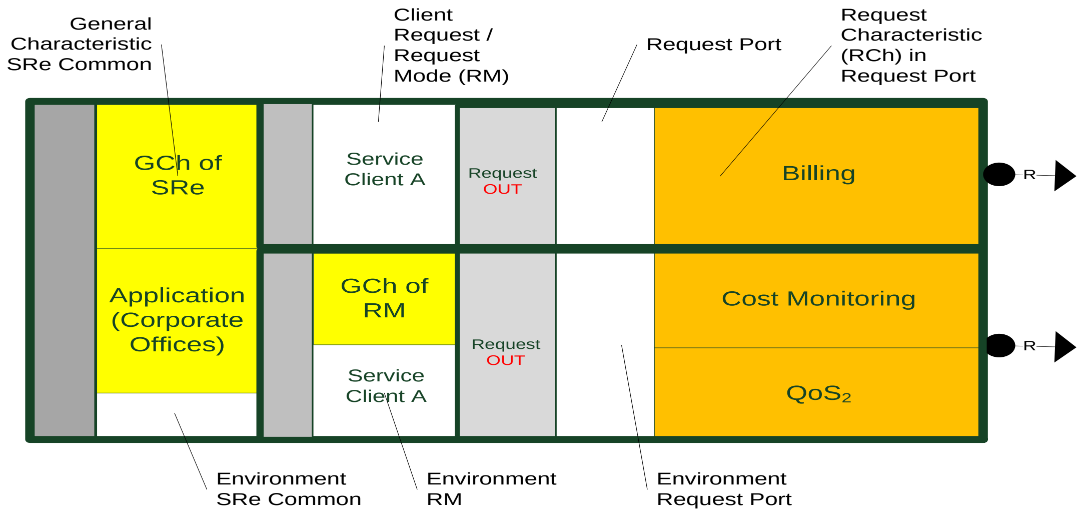


Fig. 61 - Possible Service Request for the Billing Scenario

The Request Agent will transfer the Service Request to the specifically created Composition Agent (cf. section 8.3 - Phase 3 – Service Request and Composition). It is likely that the CoA will select one of the Request Modes for the Service Client as discussed with alternative Client Requests in section 13.3 - Step2: Initialisation of the Composite Structure, in order to create the starting Actor Service Offer (ASO) of the Composite Structure (CompSt). However, since an agent is free in its pro-active behaviour, the CoA might create directly separate ASOs for both Request Modes following its own, application specific interpretation of the Service Request.

Observing the scope of this thesis, the discussion of the Service Composition by the CoA directly steps to a possible CompSt shown in Fig. 62 - Billing Scenario with resulting Composite Service. Obviously three smart grid, named from A to C, were discovered as having consumers belonging to list of consumers given with the Request Characteristic for the billing service ($\text{Billing}_{\text{RCh}}$). With the help of the Exchange Constraints of the General Characteristic for the cost monitoring service ($\text{CostMonitoring}_{\text{GCh}}$), these consumers were distributed to the fitting Service Properties of the Compatibility Characteristics ($\text{CostMonitoring}_{\text{CCh}}$) of the cost monitoring service. The Option-Slot for the Multi-port allowed the composition of three cost monitoring services. In the case of the smart grid C, a (principally) compatible Service Mode was selected, which further demands a smart metering service of a certain quality ($\text{SmartMetering}_{\text{CCh}}$, $\text{QoS}_{1,\text{CCh}}$). In the following, the discovered Facility Agents will negotiate for the grounding of the advanced service for billing. This might involve another Service Discovery and/or Service Composition done by the service environments of the Service Providers linked with the FAs. However, ACTAS certainly excluded already non-fitting Service Candidates and helped to integrate formerly separated service environments.

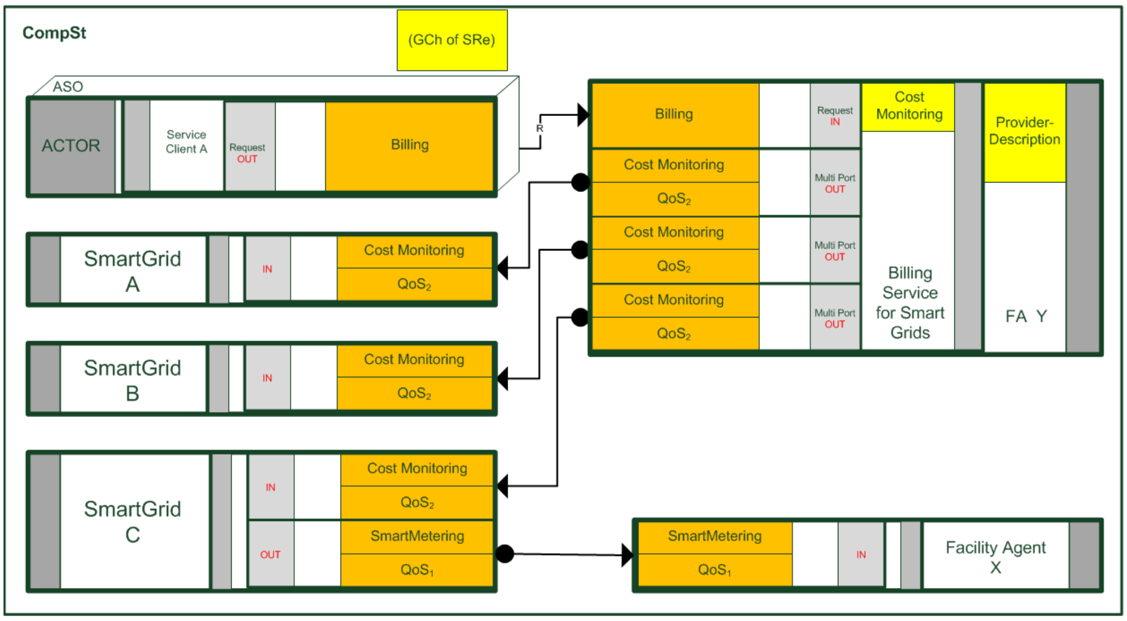


Fig. 62 - Billing Scenario with resulting Composite Service

20 Limitations of ACTAS

The section about limitations of ACTAS starts with a quote of Berners-Lee about Web Services and Semantic Web: "The argument against integration of the technologies is mainly social. It is costly to coordinate very large groups. It is much more efficient to develop WS and SW independently. Neither side has a great incentive to take on the learning required to absorb the needs and potentials of the other. Using technology in preparation by another group takes a great leap of faith, and does really add to the development time. These are real issues." [Ber2003 p. 5]

It can be no doubt, that this statement is even true in the context of ACTAS, which proposes holistic models for the categorization of services through the classification of Semantic Characteristics based on aspects of services going beyond the functional aspect. These classifications have to be introduced, published, and managed. Additionally, the models of ACTAS propose the use of established SOC algorithms in fixed semantic contexts on the abstraction level of properties of the Semantic Characteristics. ACTAS suggests that such algorithms are published and adapted, in order to use common, reliable, and actualized constraints with the applied characteristics in the descriptions of services or requests. This additional effort for the entities of ACTAS can be questioned whether justified. As limitation, it could be also discussed if ACTAS would be able to cover all relevant cases.

Berners-Lee made his statement in 2000. It is fact that the merge of Web Services and Semantic Web took already place and led to the successful approaches of Semantic Web Services. Thus, problems in various domains were answered on the base of e-services using enhanced methods of Service Description, Matching, and Composition. However, these solutions take mostly advantage of self-contained service-oriented architectures with specific repositories, interfaces, methods, and kinds of Service Descriptions. In this way, these service-oriented approaches are very well adapted to their domain but mutually often incompatible and undiscovered. With the remark of Berners-Lee in one's mind, it would be a shame not to use the expertise of the various approaches for the building of greater, comprehensive, and trans-sectoral environments.

For this purpose, an abstraction of the Service Description becomes necessary, which allows a reliable, flexible agreement on some common characteristics and appropriate constraints for the world of services, in order to discover and compose services of independent service-oriented architectures as well as to exclude non-compatible service candidates as early as possible. It is the justified object of the newly introduced entities of ACTAS, its framework character, and the pro-active behaviour of its software agents to achieve such an integration of various service-oriented application environments. As the starting quotation illustrated, the limitations come rather with the social acceptance of the additional effort. Therefore, the effort must be reasonable in terms of (1) the number of Service Modes, (2) the amount of new entities, and (3) the resulting performance of ACTAS, at all.

A Service Description in ACTAS wraps up several Service Modes. The Service Discovery process leads to the selection of one of these Service Modes. Controversially, a Service Template cannot offer Service Modes for all possible combinations of published Compatibility Characteristics relevant for an offered service, since the management of Service Modes for the Service Offers became already too vast. Nevertheless, it is not the goal of the Service Designer to cover all possible cases but to answer the likely constellations of Compatibility Characteristics. Keeping together the Service Modes of these constellations in one Service Description is an advantage on its own. In the future, the Facility Agent will pro-actively decide if he can support a requested set of CChs, i.e. the number of Service Modes in the Service Template can be automatically extended by the FA. For this purpose, the Semantic Characteristics should be semantically classified in a sufficient way.

In general, one could complain about the management of too many Semantic Characteristics and Property Classes necessary for ACTAS. Considering only the mediation between the Service Properties as realized through the Exchange Constraints, it is even more obvious that not all cases, i.e. combinations of Service Properties, can be covered. The number of possible Exchange Constraints became even exorbintant, if one would tackle all possible translations coming with the Service Properties of the Semantic Characteristics appearing in a Service Description: $(\#Chars\ x\ \#\emptyset Properties)^{\#ExNames}$ (number of involved Semantic Characteristics multiplied by the average number of their properties powered with the number of used Exchange Names, cf. Definition 18). Then again, it is not the object of environments like ACTAS or WSMO to cover every possibility. Like the mediator entities in WSMO, it is the goal to support likely cases of mediation. In ACTAS, General Characteristics can be introduced as “building blocks”, in order to ease the translation between the Service Properties of several Semantic Characteristics through Exchange Constraints.

It has to be stressed that Exchange Constraints cannot replace a programming language. Furthermore, an Exchange Constraint can just verify the information available at the time point of its application. If later value changes of some involved property objects lead to restrictions, which are incompatible with the earlier applied Exchange Constraint, then these incompatibilities might not be recognized, because an Exchange Constraint “fires” only once for performance reason (cf. section 13.6 - Step 5: Checking of Exchange Constraints). In other words, the monotony of the constraints is not directly checked by ACTAS. In this sense, ACTAS can be rather seen as a tool for the early exclusion of Service Candidates.

The number of algorithms used in the Property Classes can be reduced, when existing tools for the reasoning or interpretation are used. This was discussed in the context of the declarative FDL with a SOA-based approach published in [ShiAda et al.2010] (cf. Case Study 2: Distribute Feature Composition (DFC)). Future research will demonstrate that the creation of Semantic Characteristics can also be automated. In the case study, the integration of WSML-written ontologies, as they are listed for nfp-criteria in the appendix, through the use of the WSML2Reasoner framework was suggested. In this way, the properties described in the ontologies could be directly realised as several Char Properties or wrapped as one Char Property. Appropiate set and get methods in the related Value Property Classes could be also generated. The internal constraints defined in the ontologies would be checked by these generated algorithms through reasoning with the definitions of the ontologies (cf. rule (intOp)). Summarising, it can be said that an innovative generating of the algorithms can reduce the effort for their management extremely.

The debate about the algorithms would not be complete without a general discussion of their performance. This extended discussion can only be general since firstly more running environments of ACTAS have still to be tested, and secondly the performance of a framework middleware like ACTAS deeply depends on its current application environment. It is the strength of ACTAS, that the administrator of an application environment can adapt the Service Discovery process of ACTAS to its application through the building of pro-active software agents, i.e. a Request Agent and/or a Composition Agent. It is up to them, whether a complex Service Composition or the collection of Service Candidates is the goal of the pro-activity. In every case, the application will gain access to a set of Facility Agents, which have to do the Service Grounding. The performance of the latter, realising the Service Grounding in the perspective of ACTAS, cannot be concern of consideration in this thesis since it would depend on the discovered services and their service environments. Nevertheless, the active communication between the software agents and their pro-active behaviour should lead to an improved performance in comparison to a strict server-client approach. The algorithm of the Composition Agent is supposed to be a declarative one. It is well known, that such an algorithm can be in the worst case exponential in space as well as in time, but since it can be specialized to the application environment, one should expect a better performance. The integration of possibly non-declarative algorithms for the constraints into the declarative environment of ACTAS also leads to earlier discussed challenges (cf. section 13.1 - The Property Objects/Classes in the C-Model), which will have a huge effect on the performance, but it can be assumed that further research and standardisation will show a great improvement. The propagated use of services for the algorithms of the constraints is consequent and compulsory for an adaptive framework, but it will have its effect on the performance. This is especially true, when the algorithms include a generation of code as discussed in this section. Some hints about the slow performance in this later case can be found in [ShiAda et al.2010]. Summarizing, one could argue that the performance of ACTAS will be likely a bad one in some applications, when only used for the answering of Service Requests directly. However, the MAS of ACTAS, includes Trader Agents, which can offer appropiatly composed services. Including the Service Trading, ACTAS appears as a justified framework tool that interlinks independent service environments in a semantically reliable and commonly agreed way despite all discussed limitations.

21 Summary of evaluation

The evaluation firstly showed the rationality of ACTAS. It was identified that ACTAS is not just another Service Description environment, as they are given with WSDL for Web Services as well as OWL-S and WSMO for Semantic Web Services. It is rather a framework that combines common Semantic Characteristics of services with semantically fitting algorithms for the checking of constraints, in order to achieve a reliable Service Discovery and Service Composition. Other Service Discovery environments will not be replaced; they apply ACTAS for the advertisement of their services on a more global level through a set of commonly agreed Semantic Characteristics for the description of the interface. The MAS of ACTAS will pro-actively help to use and to integrate the discovered services. From the point of view of ACTAS, the established Service Discovery environments are necessary for the Service Grounding and Service Deployment. Therefore, it was important to clarify again the service idea of ACTAS and how this interpretation depends on its directed or non-directed interface description. The software agents of the MAS allow the support of the diverse roles of the involved parties as well as the direct consideration of non-functional parameters like the availability by the Facility Agents.

Subsequent to the general consideration of ACTAS, following topics were specifically addressed: (1) system environment, (2) service design, (3) service request, and (4) the composition process. The system environment of ACTAS will require a certain adjustment, i.e. agreements on the published Semantic Characteristics as well as the provision of established algorithms for the constraints. These algorithms must be usable within the declarative environment by ACTAS. The necessary customizations and Web Services as an approach for the provision of the algorithms were discussed. The resulting limitations coming with this approach were addressed in a separate chapter at the end of the evaluation. Through the use of related General Characteristics and Compatibility Characteristics, service descriptions in ACTAS can be assembled as from modules or building blocks. In particular the Exchange Constraints can be made available through General Characteristics. Similar to the design of Service Descriptions in ACTAS, the design of Service Requests were discussed. Based on the specific interpretation of Request Modes, the use of Alternative Client Request was mentioned. Various cases increasing in complexity were addressed in the debate of Service Composition. Among other criteria the complexity of the cases was determind through a potential existence of loops in the composition graph and the use of so-called Multi Ports. Service Ports are declared as Multi Ports through Option-Slots, when several services share a common service (e.g. a common multimedia conference facility). Another application of Multi Ports might be the use of multiple services of same type. An example in this direction could be the employment of multiple contractors for the construction of a house. Finally, the Multi Ports have been proposed for technical services such as for example the use of multiple alternative pipelines for the transport of gas.

Technical services were also content of the first of five case studies building the next part of the evaluation. Technical services include non-directed interfaces for the Service Composition.

The use of special Translation Offers was discussed. Translation Offers allow the implementation and application of technical standards, in order to examine the possible composition of technical services published with different standards. Thus, ACTAS can ensure that current standards are observed and translated during the Service Composition without the need that a service designer considers all of them in his Service Description.

The second case study evaluated the flexible application of constraints in particular for the feature compilation. It was shown that Exchange Constraints can be applied not only for the interfaces of Service Composition, i.e. Service/Request Ports, but also on a higher level of the composite structure. Additionally, it was considered how the used algorithms of the constraints can be customized through the help of the Web Services themselves. In this way, a reduction of the number of published and maintained algorithms could be achieved. Service environments like e.g. SESA for WSMO contain modules to support business policies. How the resulting inherent complexity can be supported by ACTAS, was principally discussed in the third case study.

The fourth and fifth case studies are related, since they show both the application of ACTAS and its MAS in a larger scenario. The fourth case study creates a scenario for the weather forecast, in which an assumedly existing free market of various service providers offers the provision and processing of data for the weather forecast for different geographical areas fulfilling different purposes. The adaptable offering of these services, as well as their discovery and use by applications were content of the considerations of this case study. The fifth case study enhanced the outlined ideas and applies them on an existing and evolving scenario, the so-called smart grid. After the introduction in the area of smart grids, this concluding case study shows how occurring entities and applications of a smart grid are classified at different levels through existing models of smart-grids. The service idea of ACTAS proved in this context as useful, since its categorization of services through ontologically classified Semantic Characteristics allows a perfect adaptation to these models. It was shown that the additional use of the software agents of ACTAS achieves a flexible and reliable Service Discovery and Service Composition environment, which integrates the existing environments of the smart grids.

CONCLUSION

22 Conclusion and Future Research

The experience and research in the field of distributed computing led to the development of software engineering paradigms like distributed objects, software agents, and electronic services. Early on, it was recognized that the realisation of complex information systems and technical support environments had to overcome at least two common challenges: firstly the integration of dispersed hardware and software components, and secondly an agreement on the semantic of the functional parameters. Service-Oriented Architectures gained their popularity due to the standardization and trading of Web Services addressing the challenges of integration. The addition of methods and solutions of the research area Semantic Web answered the semantic challenge of Web Services initiating approaches and standards of Semantic Web Services. An on-going goal of research is Autonomic Service-Oriented Computing, i.e. an improvement and development of adaptive solutions for the different phases of the life cycle of services, in order to reduce the necessary human intervention. Especially Service Discovery and Service Selection are decisive phases for the achievement of Autonomic SOC. The inclusion of non-functional service criteria for the Service Discovery, the federation of Service Trading, innovative methods of Service Matching and Data Mediation as well as the application of software agents for the achieving of a pro-active and re-active behaviour of service environments are extensions of SOC, in order to approach the ultimate goal of autonomic computing. However, the diverse domains led to different solutions and approaches keeping the information of their services in separate repositories. Company, government, and other policies as well as the observing of varying contexts of services for the discovery and deployment generated additionally an inherent complexity for the Service Composition and Service Coordination. Therefore, the extended challenges of Autonomic SOC demand on the one hand an adaptive classification of services, in order to avoid inflexible and incompatible repositories. On the other hand, approaches of Autonomic SOC should take advantage of the current algorithms developed for the approaches of different domain, in order to deal with the inherent complexity.

ACTAS (Adaptive Composition and Trading with Agents for Services), introduced and discussed in this thesis, is a framework for the Service Discovery and Service Composition, which complies to the made considerations of Autonomic SOC on four stages: (1) a multi-dimensional classification of services based on four aspects, (2) the checking of constraints through centralized and approved algorithms on the level of properties of the Semantic Characteristics, (3) an adaptive and context sensitive service description with the Semantic Characteristics, as well as (4) the use of software agents, in order to integrate pro- and re-actively involved environments

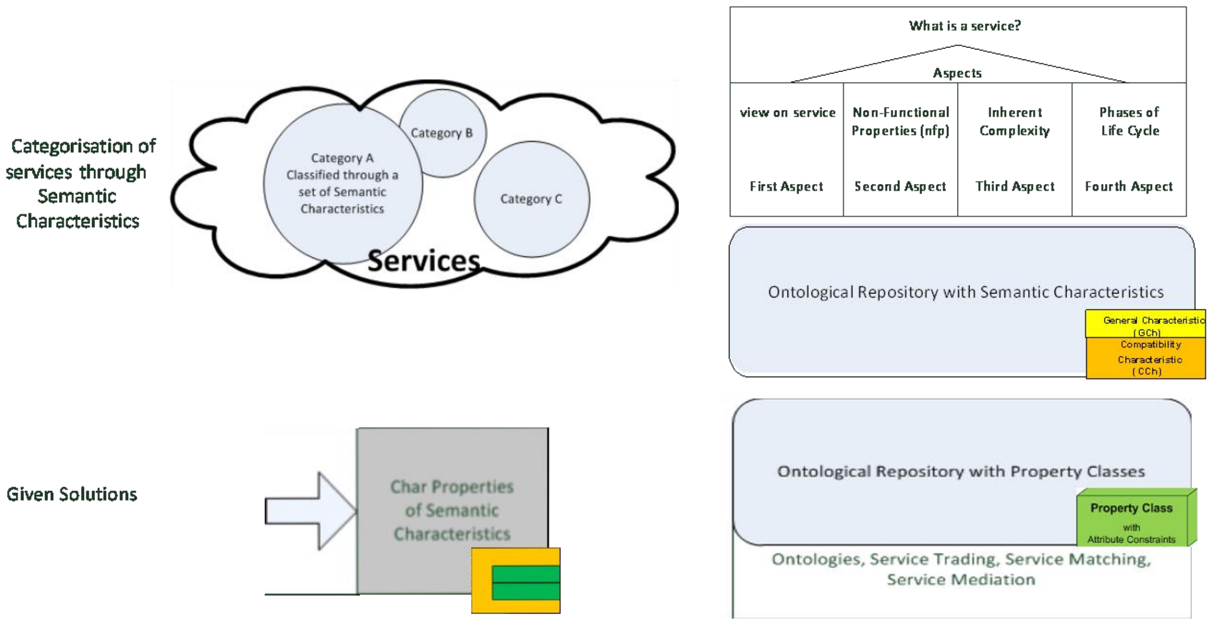

Categorisation of services through Semantic Characteristics
Category A Classified through a set of Semantic Characteristics
Category B
Category C
Services
What is a service?
Aspects
view on service
Non-Functional Properties (nfp)
Inherent Complexity
Phases of Life Cycle
First Aspect
Second Aspect
Third Aspect
Fourth Aspect
Ontological Repository with Semantic Characteristics
General Characteristic (GCh)
Compatibility Characteristic (CCh)
Given Solutions
Char Properties of Semantic Characteristics
Ontological Repository with Property Classes
Property Class with Attribute Constraints
Ontologies, Service Trading, Service Matching, Service Mediation

Fig. 63 - New entities of ACTAS

and existing solutions of SOC. The figure “Fig. 64 - ACTAS Overview” illustrates these concepts of ACTAS. The Table 26 compromises their resulting features.

Semantic Web Services use ontologically introduced semantic enhancements for the functional description of services, in order to have a reliable discovering, matching, and composition. ACTAS extends this vision to the level of the services themselves. OWL-S offers with the hierarchization of services an approach in the same direction. However, no applications of this feature of OWL-S are known. It might be due to the fact that OWL-S does not specify how this one-dimensional hierarchization between services can be achieved. It is a result of the thesis that a multi-dimensional classification of services eases the Service Discovery and Service Composition as long as commonly agreed criteria are used. Therefore, ontologically defined Semantic Characteristics can be published as a common agreement on criteria for the Service Discovery. The proposal of the four aspects of services for this purpose is a result of the thesis. Services are categorized for the Service Discovery through the association with sets of Semantic Characteristics (cf. Fig. 63 - New entities of ACTAS). The principal compatibility of services, used for the Service Discovery and Service Composition in ACTAS, is straight forward defined on the base of equal sets of Semantic Characteristics. Semantic Characteristics allow the adaptive discovery and selection of services even without functional criteria. In Appendix B, examples for ontologies of non-functional criteria are listed, which could be used for the semantic definition of Semantic Characteristics, in order to classify services for the Service Discovery through these criteria. ACTAS allows the Service Discovery not only with functional criteria. The ontologically based semantic definition of Semantic Characteristics can include criteria like application domain and policies (first and third aspect of services). Due to this inclusion of multi-dimensional criteria in the semantic definition of characteristics, separate repositories are not any longer necessary and composable services can be found easier in a more flexible way.

OWL-S offered a support for the whole life cycle of the service, whereas WSMO concentrated on its four components and kept only information about the inherent complexity of the service. The greater vision of WSMO is the awareness that ontological definitions cannot be centralized, but the algorithms for their mediation. Therefore, WSMO accesses external algorithms for the mediation. ACTAS extends this vision, when it bases its Service Selection on constraints (Value Constraints, Merge Constraints, and Exchange Constraints) using centralized algorithms that are defined externally through the introduction of Property Classes (cf. Fig. 63 - New entities of ACTAS). ACTAS is not another "complete" service environment just extending the functional description of Web Services like OWL-S or WSMO. It is a framework working with information about general e-services on two levels: (1) the Service Discovery and Service Composition based on sets of Semantic Characteristics (principal compatibility), and (2) the Service Selection through the solving of constraints associated with the properties of the Semantic Characteristics.

The WSML ontologies of non-functional criteria listed in appendix B have attributes. Such attributes can be used for the declaration of properties (Char Properties) in the semantic context of Semantic Characteristics declared for these non-functional criteria. The idea of ACTAS is to take advantage of algorithms of existing approaches. The association of Property Classes with the Char Properties allows the integration of commonly agreed, approved and secured algorithms, which can be controlled and improved due to their centralized, quasi standardized character. For instance, solutions of existing Service Discovery approaches using OWL-S, WSMO, or WSDL can be applied, when their data structures are managed in Char Properties of Semantic Characteristics, which are semantically described in an appropriate way. The fourth aspect of services covers the phases of the life cycle of a service. An appropriate semantic description of a Semantic Characteristic would use a criterion ontologically defined with the first phase of the life cycle, which is concerned with the Service Design. Further examples and appropriate semantic descriptions of Semantic Characteristics were discussed in the thesis.

In the ideal case, ACTAS can support the discovery of new Composite Services, and exclude inappropriate services at an early stage through given algorithms. For this purpose, the Service Designer can use the Semantic Characteristics like "building blocks", in order to publish alternative Service Descriptions (so-called Service Modes). A Service Requester uses the "building blocks" in a similar way for his Service Requests. The necessary data-structures of ACTAS were covered in the S-Model and R-Model. The algorithms of the Property Classes, associated with the Char Properties of the used Semantic Characteristics, can be used for a "standardized" handling of the introduced service/request properties under the consideration of the semantic context given with the used Semantic Characteristic as well as specific ontologies and Value Constraints valid in the context of the Service Mode or Service Request, respectively. Thus, the context of the service or the request can also be involved in the processing of ACTAS relying on established ontologies and algorithms for the constraints.

ACTAS is kept simple; it works with only two kinds of Semantic Characteristics (General Characteristic and Compatibility Characteristic) and three different kinds of constraints (Value, Merge, and Exchange Constraints). The Compatibility Constraints are used for the description of compatibility between services as well as service and request. The principal compatibility leads to Comparable Properties. The Merge Constraints built for these Comparable Properties allow the application of established matching and mediation algorithms. The General Characteristics support the Service Selection through Value Constraints. The usages of Exchange Constraints, which import algorithms dealing with properties of possibly several Semantic Characteristics, were also discussed in the thesis. An extended mediation defined for several Service/Request Properties was examined with the introduction of so-called Translation Offers.

In the fourth stage of the support of the vision of Automatic SOC by ACTAS, the thesis introduces a Multi-Agent System (MAS) environment, which integrates the application, trading and provision environments through specific agents. The S-Model of ACTAS supports the agents responsible for the Service Provision interface (Facility Agent, FA) in the management of the resources, negotiation, and deployment of services (e.g. distinction between ST and SOER). The Service Grounding and Service Deployment of ACTAS are done through the discovered and selected Service Provision. These phases are likely to integrate existing Service Oriented Architectures dealing with the inherent complexity of the Composite Service. The integration of an existing SOA was also discussed in the context of the trading environment using Semantic Characteristics, which have a special trading criterion in their semantic description.

The dealing with a Service Request was introduced with the C-Model and evaluated through several scenarios debating services in technical and business domains. The application environment is the origin of the Service Request and it is also in control of the Service Composition process as it generates the responsible Composition Agent (CoA). The Service Composition Process takes place in a declarative environment. Therefore, the advantages of such environments can be used for the Service Selection involving the context of the services and established algorithms. The challenges of the integration of centralized algorithms into the declarative environment of ACTAS through Property Objects, built from the Property Classes, were discussed in the thesis. Concern of future research will be the centralized provision of Property Classes and their deployment as Property Object, in order to reassemble the stack of the declarative environment.

A sophisticated support for Service Designer and Service Requester is a goal of future research. Prototypes of the agents shall help with the creation of appropriate agents for the MAS of ACTAS. A tool may assist with the Service Description and the building of Service Request. It would show the available Semantic Characteristics. The “works-with” relationship between the Semantic Characteristics will be the basis for such kind of a tool. This relationship will also be helpful for the dynamic creation of Service Modes and Service Requests done by the agents during the Composition Process. Like a Service Description can have several Service Modes, a future Service Request could deal with alternative Client Request as discussed in the thesis. The learning of Service Client preferences with the help of Personal Agents and Actor Service Templates (AST), which could be compared with Service Templates of the Service Offers, is another intention of future research. ACTAS enables the involvement of existing environments of Service Oriented Computing in several ways. The later phases of the life cycle of services in ACTAS are built on their expertise. The extension of the composition process with such environments especially for Service Trading can be controlled through suitable semantic descriptions of the Semantic Characteristics. Therefore, the concepts of ACTAS can become an exciting step towards Autonomic Service Oriented Computing.

Feature of ACTAS	Description
Mapping of Algorithms Char Property Class Merge Property Class Exchange Property Class	• Property Classes realize access to algorithms • The declaration of properties as Char Properties in the context of Semantic Characteristics maps their algorithms and settings into this semantic context. In this way, specific constraints and ontologies can be set. • Algorithms are distinguished between algorithms for the management of properties (Char Property Classes), for the matching of properties (Merge Property Classes), and for the mediation of several properties (Exchange Property Classes).
Getting information for the Negotiation Phase	• The declarative solving of constraints through approved and context sensitive selected algorithms leads to new information useful for the Negotiation Phase. • The selection of a certain Service Mode in the Composition Process with the rules of principal compatibility, early determines the direction of the negotiation. • Reservation of resources or the Component Services on the base of the SOER could make the subsequent Deployment Phase more reliable.

Feature of ACTAS	Description
Adaptation for different domains and users (1st aspect of services: the view on the service) Consideration of other aspects of service (2nd aspect: non-functional aspects) (3rd aspect: inherent complexity) (4th aspect: phases of the life cycle)	• The Semantic Characteristics define Value and Exchange Constraints, in order to describe constraints, which are relevant the information of its Char Properties in its semantic context, e.g. domain specific restrictions for the values and ontologies could be specified. • Semantic Characteristic can also wrap properties describing other aspects of services more closely, e.g. non-functional aspects like trust and reliability. • Request Characteristics, a specific kind of Compatibility Characteristics, can be related with designated user groups, in order to precise the Service Requests. • Each Semantic Characteristic classifies through its Semantic Description services, i.e. services fulfilling the constraints of the Semantic Description are addressed. The combination of Semantic Characteristics means an intersection of the sets of the addressed services of each Semantic Characteristic.
Simple Description of Service Compatibility	• Principally Compatible Component Services are found through common sets of Compatibility Characteristics (a specific kind of Semantic Characteristic) in their interface (i.e. Service Port). • Differentiation between B2C and B2B Service Composition through the use of solely Request Characteristics in the interface description of a service. • The compatibility of services is checked on the level of Service Properties through Merge Constraints, which test the matching of Service Properties, which became comparable due to the principal compatibility of their services. The Merge Constraints allow the integration of approved matching and mediation algorithms.

Feature of ACTAS	Description
Access of implementation instances of the algorithms associated with the Service Properties through Property Objects and their integration in a declarative environment	• Property Objects are created on the base of Property Classes • They encapsulate the handle for the access of an implementation instance of the algorithm described with the Property Class. • The clowning of the Property Objects and their implementation instances as well as the addressing of variants enables the backtracking of the declarative environment of the C-Model, in order to realize the Composition Process. That means that the implementation instances have to support their clowning as well as that their methods should support the addressing of variants. • Service Properties are declared as Char Properties with Char Property Classes in the context Semantic Characteristics. Their Property Objects are created, when the Service and Request Descriptions of the S-Model and R-Model are used in the declarative environment described through the C-Model. • Service Properties, which were declared as Char Properties in the context of a Compatibility Characteristic, are additionally associated with a Merge Property Class. A Merge Property Object will be built, when the Merge Constraint is applied on this Service Property and a comparable one. • The mediation between several Service Properties is done with Exchange Constraints, which use methods of Exchange Property Classes. The Exchange Property Objects are built, when the methods are applied. • For the access of Service Properties of Merged Service Ports, the Exchange Constraints use Merge Property Objects, in order to receive Property Objects and in order to hand them back. In this process, the Merge Property Object will check if the information of the Property Object still complies with the Merge Constraint.

Feature of ACTAS	Description
Multi-Agents System (MAS) Facility Agent (FA) Personal Agent (PA) Trader Agent (TrA) Request Agent (ReA) Composition Agent (CoA)	• The MAS is the running environment, the middleware for the declarative environment of ACTAS. It allows the concentration on the Service Discovery based on the Semantic Characteristics. The other environments of (autonomic) Service Processing can be addressed by the ReA subsequently. The gained information of ACTAS can help to address the right environments and Service Providers. • The agents are pro-active, running algorithms, which are determined through the roles involved in the Service-oriented Architecture. The FA realises the policies of the Service Provider. The ReA is part of the application environment. Therefore, it will create a CoA for a Service Request, which performs a Composition Process according to the policies of the application. Each TrA follows its own policies in providing Service Offers. The PA introduces the interest of the additional role of a Service Client, which is necessary for a commonly usable framework like ACTAS. • The agents can also support the subsequent phases of the life cycle of services. The Facility Agents can play an active role in the negotiation phase. The feedback of the Execution Phase could be used for a redo of the Composition Process or the learning of users' preferences. • The FA could realise the resource management including the reservation of resources for a Component Service. The Res-Info field in the Selected Service Mode of the Composite Structure (CompSt) (cf. **Definition 24**) could support the FA. • The FA can react on a changed availability with new SOERs.

Table 26 - Features of ACTAS

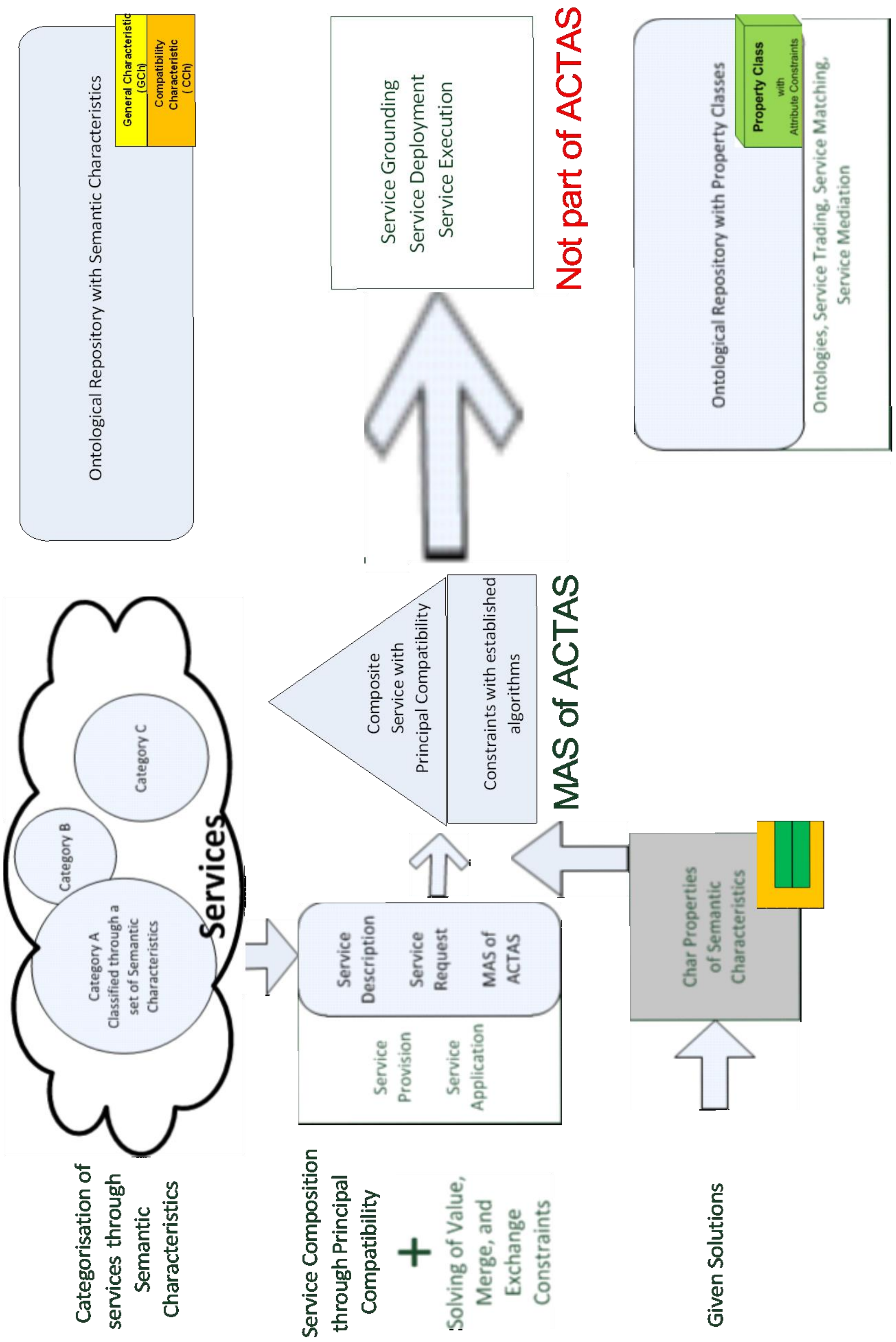

Categorisation of services through Semantic Characteristics
Services
Category A Classified through a set of Semantic Characteristics
Category B
Category C
Ontological Repository with Semantic Characteristics
General Characteristic (GCh)
Compatibility Characteristic (CCh)
Service Composition through Principal Compatibility
+
Solving of Value, Merge, and Exchange Constraints
Service Provision
Service Application
Service Description
Service Request
MAS of ACTAS
Composite Service with Principal Compatibility
Constraints with established algorithms
MAS of ACTAS
Service Grounding
Service Deployment
Service Execution
Not part of ACTAS
Given Solutions
Char Properties of Semantic Characteristics
Ontological Repository with Property Classes
Property Class with Attribute Constraints
Ontologies, Service Trading, Service Matching, Service Mediation

Fig. 64 - ACTAS Overview

APPENDIX

A Collection of used ontologies and ACTAS Entities

In this chapter of the appendix the used ontologies and entities of ACTAS in the thesis are listed. The listing of ACTAS entities begins with the various Property Classes. Merge Property Classes are normally marked with “-me”, whereas Exchange Property Classes have the mark “-ex”. Only methods and properties, mentioned in the text, are part of the listing.

A-1 Used Ontologies

Various ontologies are developed and will be developed that can be used as classification additions for the characteristics of ACTAS. For the semantic description of the Semantic Characteristics, the main ontologies should be related to the four aspects of services introduced in the State-of-the-Art. For simplification, ontologies for the direct support of the aspects of services shall be assumed in this thesis. In the chapter B of the appendix, existing ontologies for the support of the second aspect, the non-functional properties (nfp) are listed.

Assumed Ontology	Description	
$\text{Ontology}_{\text{Domain}}$	Concepts of application domains	(1st aspect of services)
$\text{Ontology}_{\text{nfp}}$	Non-functional concepts (cf. also chapter B)	(2nd aspect of services)
$\text{Ontology}_{choreography}$	Concepts for the choreography	(3rd aspect of services)
$\text{Ontology}_{\text{Design}}$	Concepts for the designs E.g. three kinds of language concepts for Service Descriptions are distinguished: written language (e.g. English), spoken language (e.g. German) and standard language (e.g. WSDL).	(4th aspect of services)
$\text{Ontology}_{\text{trading}}$	Concepts to the Trading Phase	(4th aspect of services)
$\text{Ontology}_{matching}$	Concepts to the Matching Phase	(4th aspect of services)
$\text{Ontology}_{\text{Cycle}}$	Concepts for the other phases of Life Cycle	(4th aspect of services)
$\text{Ontology}_{\text{User}}$	Concepts for the classification of users	(used by RCh)

Table 27 - Assumed Ontologies for the classification

A-2 ACTAS entities

In the following subsections ACTAS entities used in the thesis are listed:

- General Characteristics (GCh)
- Compatibility Characteristics (CCh), including the Request Characteristics (RCh)
- Property Classes

A-2.1 Descriptions of used General Characteristics (GCh)

General Characteristic (GCh)	Description
Audio-H.323	Used in Example 9. A General Characteristic, which is supposed to translate between the Service Properties of the following CChs: H.323 and Audio-Com. This translation shall be similar to the one sketched in Example 15.
Audio-Phone	Used in Example 15 A General Characteristic, which is supposed to translate between the Service Properties of the following CChs: Phone and Audio-Com.
AV-Com (RCh)	Used in Example 9. Communication service for audio and video communication
AV-Com (RCh)	Used in Example 9. Communication service for audio and video communication
AV-Conference (RCh)	Used for the demonstration of Multi Port in Fig. 41. It belongs to the domain of telecommunication. No Char Properties given.
AV-H.323	Used in Example 9. A General Characteristic, which is supposed to translate between the Service Properties of the following CChs: H.323 and AV-Com. This translation shall be similar to the one sketched in Example 15.
AV-H.323-Reliability	Used in Example 9. This General Characteristic is supposed to integrate Exchange Constraints, which enable the checking of reliability and mediate between the data of the Service/Char Properties of the CCh H.323-Reliability and the RCh AV-Reliability. This means, they translate the requested reliability data of the user into the data of the H.323 standard and its reliability description.
Feature	This General Characteristic includes Exchange Constraints for the checking of the choreography of message sequences generated on the basis of a set of telephone features (cf. Case Study 2: Distribute Feature Composition (DFC)).
Feature-Composition	This General Characteristic is an example for the use of General Characteristics in the Common Part of the Service Request. It offers Exchange constraints for the checking of Distributed Feature Constraints of two sets of features located in the Client Request. The set of features are described with the declarative FDL language debated in Case Study 2: Distribute Feature Composition (DFC).
Planning	Used in Fig. 41 - Composite Structure (CompSt) with use of Multi Ports The Request Characteristic "Loc-Auth" (cf. equation (22-16)) is used, in order to clarify that the MCU is reachable at a time slot and the costumer has an authorisation for the using of the MCU. The General Characteristic "Planning" could verify the planning of the time slots of the Service Clients.

General Characteristic (GCh)	Description
Provider-Description	Used in Example 9 In the appendix Chapter B, the nfp-ontologies are listed. The provider ontology is one of these. In the thesis, it is described how these ontologies given in WSML can become Semantic Characteristics of ACTAS using existing WSML interpreter for the implementation of the Property Classes (cf. 14.1 - Environment of ACTAS).
Reliability	Used in Example 9. This General Characteristic might work with other General Characteristics or Compatibility Characteristics about reliability. A good candidate is the Request Characteristic of the same name, which might deliver values based on the same standard built for the Component Services. Its Exchange Constraints could be linked with Char Properties of these characteristics, in order to calculate a more general value for the reliability of the service, based on several reliability values originated from the other characteristics.

Table 28 - Used General Characteristics (GCh)

A-2.2 Descriptions of used Compatibility Characteristics (CCh and RCh)

Compatibility Characteristic (CCh) (CChs declared as RCh are marked)	Description
Availability (RCh)	Discussed in chapter 14. The nfp-ontology for availability is shown in section B-1 of the appendix. The ontology defines several properties for the concept "Availability": "isAvailableAt", "isAvailableDuring", and "isAvailableTo". This concept is extended to the concept "RequestAvailability" providing the additional properties: "forRequest", "hasNegotiableTime", and "isContinuouslyAvailable".
AV-Com (RCh)	Used in Example 9. This Request Characteristic demands a service, which supports Audio-Video Communication. Its Char Properties shall not be too technically, since it describes a user interface.
AV-Com (RCh)	Used in Example 9. Communication service for audio and video communication
AV-Conference (RCh)	Used for the demonstration of Multi Port in Fig. 41. It belongs to the domain of telecommunication. No Char Properties given.
AV-Reliability (RCh)	Used in Example 9. The existence of this Request Characteristic was motivated through the assumption that a standard for AV connection exist, which are open to the Service Client. That means that Service Client can and will demand services, which support this standard. ACTAS allows the discovery of AV-services supporting just this standard through the combination with the RCh AV-Com. Other possible incompatible standards of AV might have other Request Characteristics. Thus, this characteristic shows the adaptive pre-selection through categorisation in ACTAS pretty nicely.

Compatibility Characteristic (CCh) (CChs declared as RCh are marked)	Description
General-Com (RCh)	It is used in the case study 1, in order to discuss the application of Translation Offers, i.e. the autonomic translation between the RChs: AV-Com, Audio-Com, Written-Com, and General-Com. This RCh allows the discovery of communication services providing any kind of communication: written, audio, video or combinations. Its Char Properties shall not be too technically, since it describes a user interface.Properties: Transfer – Audio, Video, Written or any combination of them
GeodataWSMOTrader	The Semantic Description of this CCh in equation (9-1) in section 9.1 - S-Model: Semantic Characteristics - motivated the idea of Example 5 that a trading criteria associated in the semantic description with a Semantic Characteristic can be interpreted through the Trader Agents of ACTAS and even include external trading environments in this way. Transfer – Audio, Video, Written or any combination of them
H.323	Used in Example 9. Non-directed Technical Service for communication facilities fulfilling the H.323 standard for video-conferencing , i.e. communication based on audio and video transfer (av)
H.323-Reliability	Used in Example 9. This Compatibility Characteristic is supposed to wrap some Char Properties, which keep some information about the reliability of H.323 connections. This Semantic Characteristic can be combined with the H.323 characteristic, in order to look for H.323 connections, which support the reliability rules connected with this CCh.
Insurance (RCh)	Used in Example 13 In the semantic description, this Semantic Characteristic is associated with the domain “TravelInsurance”. Therfore, a combination with the RCh “Travel” will look for services, which allow a booking of travel and offer also a fitting insurance.
Location (RCh)	The semantic description of the CCh “Location” in equation (22-17) of Table 34 - Request Characteristics (RCh) is defined through the nfp-ontology concept “GeographicalRegion” (cf. appendix section B-3 - NFP-Ontology for location (locative)).
Loc-Auth (RCh)	Used in Example 9 Allows the identification of the location of a service. It can be used for the checking of the security, e.g. the authentification of the user. It might also contain Char Properties for the checking of the availability of the service, in order to do even some planning.
Phone	Used in several examples and case study 1 The CCh is supposed to describe the connection between telephone exchange/switchboards. These are non-directed Technical Services, which shall stay transparent for the Service Clients, the telephone customers. Therefore, the CCh is not declared as RCh.
Pipeline	This CCh demonstrates several compositions between two selected Service Modes (cf. section 14.4 - Composition Process).

Compatibility Characteristic (CCh) (CChs declared as RCh are marked)	Description
Reliability (RCh)	This General Characteristic might work with the General Characteristic of the same name. It works with the same standards of reliability like the GCh. As a Request/Compatibility Characteristic it can deliver reliability values of the Component Services supporting these reliability standards.
Travel (RCh)	Used in Example 13 This Request Characteristic shall allow the search for services providing the booking of travels.

Table 29 – Used Compatibility and Request Characteristics

A-2.3 Property Classes

In the following tables, assumed Property Classes are listed and described as they are mentioned in the thesis. Their preferred methods are sketched. The listing is separated in two tables. The first one enumerates Char and Merge Property Classes. The second table lists the sketched Exchange Property Classes.

The Char and Merge Property Classes are described together, since them both part of an association with a Char Property in a Compatibility Characteristic at declaration time. The Merge Property Class just joins Service Properties of the same Char Property Class through the "merge" method (cf. C-Model, section 13.5 - Step 4: Checking of Merge Constraints). It further offers methods for the "borrowing" and "returning" of a property of that Char Property Class, in order to enable the application of an Exchange Constraint (cf. C-Model, section 13.6 - Step 5: Checking of Exchange Constraints).

In the thesis, it is proposed that the Char Property Classes should contain methods for accessing its information especially for the later phases of the life cycle of the discovered Composite Services of ACTAS. Beside others, the method "printValues" was suggested (cf. page 86).

Char Property Class	Description	Used for Char Properties in Char
audioQuality audioQuality-me	Expressing the quality of audio transport in a user friendly way. In Example 15, its value is mediated to values of properties of the CCh "Phone".	Audio-Com (RCh)
availability availability-me	With a WSDL interpreter as discussed in section 14.1, the implementation instances of these Property Classes could be realized, since the declaration of the Request Characteristic "Availability" follows an nfp-ontology written in WSML. In the declaration of the RCh "Availability", all three given Char Properties are declared with this Property Classes, in the assumption, that they have methods for the initialisation fitting to the actual property.	Availability (RCh)
declarativeFDL declarativeFDL-me	The Property Class "declarativeFDL" can interpret declarative programs describing the constraints between diverse features. The implementation instance could be Web Services following ideas of SOA based support for this DSL language published in [ShiAda et al.2010].	Feature (RCh)
sequenceFDL	The two Property Classes "sequenceFDL" and "declarativeFDL" are introduced in the case study about the solving of constraints in the context of DFC. Two languages, named Feature Description Languages (FDL), were described. The Property Class "sequenceFDL" shall support a FDL, which describes the features as message sequences between phone and exchange.	Feature (GCh)
policySpec policySpec-me	In Example 13 Merge Constraints are motivated with the listed Char and Merge Property Classes.	Insurance (RCh)
OWL-S-capability OWL-S-capability-me	Introduced in Example 7 as a Char Property Class for the handling of capability information inside of an OWL-S description. It has a method for the initialisation with an OWL-S description referenced through a URL. Ideally, the class include a plausibility check for the OWL-S description. The Merge Property Class realises the Merge Constraint (method "merge") checking the IOPE of the comparable properties declared with the Char Property Class OWL-S-capability. The direction of the composition is important, since only in this way the client and server description in OWL-S can be distinguished.	OWL-S_with_IOPE (RCh)

Char Property Class	Description	Used for Char Properties in Char
phoneSpeed phoneSpeed-me phoneQuality phoneQuality-me providerSpec providerSpec-me	Two Property Classes are used for the sketching of the technical "Phone" service in the thesis. In Example 15) the information of Service Properties declared with theses Property Classes is translated to the information of the RCh "Audio-Phone" and vice versa. In Example 13 Merge Constraints are motivated with the listed Char and Merge Property Classes.	Phone (CCh)
journeySpec journeySpec-me costSpec costSpec-me	In Example 13 Merge Constraints are motivated with the listed Char and Merge Property Classes.	Travel (RCh)

Table 30 – Used Char Property Classes

Exchange Property Class	Description	Used for Exchange Constraints in
audioQuality-ex	Offers methods for the mediation/translation between the RCh "Audio-Com" and the CCh "Phone". An example for B2C mediation in ACTAS (cf. Case Study 1: Technical Services with translation).	Audio-Phone
declarativeFDL-ex	Offers methods for the checking of Distributed Feature Constraints of two sets of features described with the declarative FDL language debated in Case Study 2: Distribute Feature Composition (DFC).	Feature-Composition
phoneH323-ex	Offers methods for the mediation between Char Properties of the Semantic Characteristics "H.323" and "Phone", which describe Technical Services, which shall be kept transparent towards the Service Clients. It is an example of B2B mediation in ACTAS (cf. Case Study 1: Technical Services with translation).	Phone-H.323
sequenceFDL-ex	Offers methods for the choreography between telephone exchange and phone set based on message sequences generated on the basis of a set of telephone features (cf. Case Study 2: Distribute Feature Composition (DFC)).	Feature (GCh)

Table 31 – Used Exchange Property Classes

A-3 Formal definitions of Semantic Characteristics

In this section, the Semantic Characteristics used in the thesis are shortly described. The declarations of the Semantic Characteristics are listed in separate tables in the following sub-sections.

In the context of the use of Semantic Characteristics as "building blocks", the relation "Works-with" was mentioned (cf. Fig. 28 - Principal ontological categorization of Semantic Characteristics). The relation allows the Service Designer to find Semantic Characteristics, which are related to each other. This relation can be a translation/mediation of values as shown in Example 15. It can also be the testing of further going constraints between the involved Service Properties like in Case Study 2: Distribute Feature Composition (DFC). The "Works-with" relation has the subsequent signature:

Works-with (<Char1>, <Char2>, <Ex-CoRef>)

Char2, which is a General Characteristic normally, contains an Exchange Constraint referenced through Ex-CoRef. The relation expresses that the referenced Exchange Constraint could be used with a Service/Char Property declared in the Semantic Characteristic Char1. The used Service Properties of an Exchange Constraint are listed in the "ExchangeProperties" term (cf. Definition 18). In so-called pre-defined Exchange Constraints only the Property Classes and the preferred views are listed in the "ExchangePropertties" term. The "ExchangeProperties" term can be adapted through the "exchangeProperties" Option-Slot (cf. Table 14 - Option-Slots of Service Modes and Common Part of Service Descriptions). This was shown in Example 15 and in several case studies of the thesis.

In the ontology of characteristics, the concept of an RCh has a specific relationship to concepts of a user ontology (in Fig. 28 - Principal ontological categorization of Semantic Characteristics: „Can_be_used_with"-relation), i.e. the semantic description (SemDescr in Definition 4 and Definition 5) of a Request Characteristic restricts the semantic context of Client Requests to specific user groups.

Like the Service Designer does not have to use every Char Property of the Semantic Characteristics, since ACTAS is an open and adaptive environment, the declarations in this section cannot be complete, either. In the header of the tables the definition of the entities is partly repeated. The elements of the sets are simply listed. For simplification their indices start with one every time as long as the listing is unambiguous.

A general remark to the terms Char Property and Service Property shall be repeated. A Semantic Characteristic wraps Char Properties. When a Semantic Characteristic gets part of a Service Description (or Service Request) then its properties will be also addressed as Service Properties.

A-3.1 General Characteristics

General Characteristics (GCh $\equiv$ (**GCh**$^{\mathbf{ID}}$**, SemDescr, GCh_Property**$^{\mathbf{Set}}$**, Char_Env***))* **Char_Env** $\equiv$ **(Char_Va_Co**$^{\mathbf{Set}}$**, Char_Ex_Co**$^{\mathbf{Set}}$**)**		
Provider-Description	SemDescr = $(\{GCh\}, [(\text{Ontology}_{nfp}, \text{ProviderDescription})])$	(22-1)
Reliability	SemDescr = $(\{GCh\}, [(\text{Ontology}_{nfp}, \text{CompositeService\_Reliability})])$	(22-2)
Audio-Phone	SemDescr = $(\{GCh\}, [(\text{Ontology}_{domain}, \text{telecommunication}) \sqcap (\text{Ontology}_{tele}, \text{phone})])$	(22-3)
	Char_Ex_Co$_1$ = ex-co$\left(Ex_Co_1^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, AQ), (\text{PC}_2, \text{View}_2, S), (\text{PC}_3, \text{View}_3, Q)]),\ classes([\text{audioQuality-ex}])\right), \left([\text{audioQuality-ex.hasValue}([AQ], [\ \ ])] \Rightarrow [\text{audioQuality-ex.translation_phone}([AQ, S, Q], [max])]\right)\right)$	

	Works-with(Audio-Com, Audio-Phone, Ex_Co1)
	Works-with(Phone, Audio-Phone, Ex_Co1)	

Audio-H.323	SemDescr = $(\{GCh\}, [(\text{Ontology}_{domain}, \text{telecommunication}) \sqcap (\text{Ontology}_{tele}, \text{H.323})])$	(22-4)
	Char_Ex_Co$_1$ = ex-co$\left(Ex_Co_1^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, AQ), (\text{PC}_2, \text{View}_2, HQ)]),\ classes([\text{audioQuality-ex}])\right), \left([\text{audioQuality-ex.hasValue}([AQ], [\ \ ])] \Rightarrow [\text{audioQuality-ex.translation_h323}([AQ, HQ], [max])]\right)\right)$	

	Works-with(Audio-Com, Audio-H.323, Ex_Co1)
	Works-with(H.323, Audio-H.323, Ex_Co1)	

Phone-H.323	SemDescr = $(\{GCh\}, [(\text{Ontology}_{domain}, \text{telecommunication}) \sqcap (\text{Ontology}_{tele}, \text{phone}) \sqcap (\text{Ontology}_{tele}, \text{H.323})])$	(22-5)

	Char_Ex_Co$_1$ = ex-co$\left(Ex_Co_1^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, PQ), (\text{PC}_2, \text{View}_2, HQ)]),\ classes([\text{phoneH323-ex}])\right), \left([\text{phoneH323-ex.hasValue}([PQ], [\ \ ])] \Rightarrow [\text{phoneH323-ex.translation}([HQ, PQ], [])]\right)\right)$
	Char_Ex_Co$_1$ = ex-co$\left(Ex_Co_2^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, PQ), (\text{PC}_2, \text{View}_2, HQ)]),\ classes([\text{phoneH323-ex}])\right), \left([\text{phoneH323-ex.hasValue}([HQ], [\ \ ])] \Rightarrow [\text{phoneH323-ex.translation}([HQ, PQ], [])]\right)\right)$	

	Works-with(H.323, Phone-H.323, Ex_Co1)
	Works-with(Phone, Phone-H.323, Ex_Co1)
	Works-with(H.323, Phone-H.323, Ex_Co2)
	Works-with(Phone, Phone-H.323, Ex_Co2)	

Feature	SemDescr = $(\{GCh\}, [(Ontology_{domain}, \text{telecommunication}) \sqcap (\text{Ontology}_{\text{Choreography}}, \text{simpleMessageExchange})])$	(22-6)
	GCh_Property$_1$ = (SequenceDiagram, sequenceFDL)	

General Characteristics $(GCh \equiv (\mathbf{GCh^{ID}}, \mathbf{SemDescr}, \mathbf{GCh\_Property^{Set}}, \mathbf{Char\_Env}))$ $\mathbf{Char\_Env} \equiv (\mathbf{Char\_Va\_Co^{Set}}, \mathbf{Char\_Ex\_Co^{Set}})$		
	$\text{Char\_Ex\_Co}_1 = \text{ex-co}\left(Ex\_Co_1^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, Features), (\text{PC}_2, \text{View}_2, Sequence)]), classes(\text{sequenceFDL} - \text{ex})\right), ([\ \ ] \Rightarrow [\text{sequenceFDL-ex.checkSeq}([Features, Sequence], [\,])])\right)$	
	Works-with(FeatureCCh, FeatureGCh, Ex_Co1)	
Feature Composition	$\text{SemDescr} = (\{\text{GCh}\}, [(\text{Ontology}_{domain}, \text{telecommunication})])$	(22-7)
	$\text{Char\_Ex\_Co}_1 = \text{ex-co}\left(Ex\_Co_1^{ID}, \left(properties([(\text{PC}_1, \text{View}_1, FeatureOne), (\text{PC}_2, \text{View}_2, FeatureTwo)]), classes(\text{declarativeFDL} - \text{ex})\right), ([\ \ ] \Rightarrow [\text{declarativeFDL-ex.checkDFC}([FeatureOne, FeatureTwo], [\,])])\right)$	

Table 32 - General Characteristics (GCh)

A-3.2 Compatibility Characteristics

Compatibility Characteristics $(\mathbf{CCh} \equiv (\mathbf{CCh^{ID}}, \mathbf{SemDescr}, \mathbf{CCh\_Property^{Set}}, \mathbf{Char\_Env})$ $\mathbf{Char\_Env} \equiv (\mathbf{Char\_Va\_Co^{Set}}, \mathbf{Char\_Ex\_Co^{Set}})$		
Phone	$\text{SemDescr} = \left(\{CCh\}, \left[(Ontology_{domain}, telecommunication) \sqcap (\text{Ontology}_{tele}, \text{telestandard})\right]\right)$	(22-8)
	$\text{CCh\_Property}_1 = (\text{Speed}, \text{phoneSpeed}, \text{phoneSpeed-me})$ $\text{CCh\_Property}_2 = (\text{Quality}, \text{phoneQuality}, \text{phoneQuality-me})$ $\text{CCh\_Property}_3 = (\text{Provider}, \text{providerSpec}, \text{providerSpec-me})$	

	Works-with(Phone, Audio-Phone, Ex_Co1)
	 Works-with(Phone, Phone-H.323, Ex_Co1)
	 Works-with(Phone, Phone-H.323, Ex_Co2)	

H.323	$\text{SemDescr} = \left(\{CCh\}, \left[(Ontology_{domain}, \text{telecommunication}) \sqcap (\text{Ontology}_{tele}, \text{telestandard})\right]\right)$	(22-9)

	Works-with(H.323, Audio-H.323, Ex_Co1)
	 Works-with(H.323, Phone-H.323, Ex_Co1)
	 Works-with(H.323, Phone-H.323, Ex_Co2)	

Table 33 - Compatibility Characteristics (CCh)

A-3.3 Compatibility Characteristics for Service Requests – Request Characteristics

Request Characteristics (RCh) (like CCh but additionally the relation "Can_be_used_by" (Services, Users))		
Audio-Com	SemDescr = ({RCh}, [$(Ontology_{domain}$,telecommunication) $\sqcap$ $(\forall Can\_be\_used\_by$.all)])	(22-10)
	$\text{CCh\_Property}_1 =$ (Audio-Quality, $audioQuality, audioQuality\_merge$)	
	Works-with(Audio-Com, Audio-Phone, Ex_Co1)	
Availability	SemDescr = ({RCh}, [$(Ontology_{\text{nfp}}$,availability) $\sqcap$ $(\forall Can\_be\_used\_by$.all)])	(22-11)
	$\text{CCh\_Property}_1 =$ (isAvailableAt, availability,availability-me) $\text{CCh\_Property}_2 =$ (isAvailableDuring, availability,availability-me) $\text{CCh\_Property}_3 =$ (isAvailableTo, availability,availability-me)	
AV-Com	SemDescr = ({RCh}, [$(Ontology_{domain}$,telecommunication) $\sqcap$ $(\forall Can\_be\_used\_by$.all)])	(22-12)
	Works-with(AV-Com, General-AV, Ex_Co1)	
AV-Conference	SemDescr = ({RCh}, [$(Ontology_{domain}$,telecommunication) $\sqcap$ $(\forall Can\_be\_used\_by$.all)])	(22-13)
Domain-Geodata	SemDescr({RCh}, [(Ontology$_{\text{Domain}}$,Geodata)])	(22-14)
Feature	SemDescr = ({RCh}, [$(Ontology_{domain}$,telecommunication) $\sqcap$ $(\forall Can\_be\_used\_by$.phoneCustomer)]) $\text{CCh\_Property}_1 =$ (Depedency, declarativeFDL, declarativeFDL-me)	(22-15)
Loc-Auth	SemDescr({RCh}, [(Ontology$_{\text{nfp}}$, GeographicalRegion) $\sqcap$ (Ontology$_{\text{nfp}}$, Availability) $\sqcap$ (Ontology$_{\text{nfp}}$, Security)])	(22-16)
	$\text{CCh\_Property}_1 =$ (RegionSpec, audioQuality, audioQuality-me)	

Request Characteristics (RCh) **(like CCh but additionally the relation "Can_be_used_by" (Services, Users))**		
Location	$\mathrm{SemDescr}\left(\{\mathrm{RCh}\}, \left[\left(\mathrm{Ontology}_{\mathrm{nfp}}, \mathrm{GeographicalRegion}\right)\right]\right)$	(22-17)
OWL-S _with_IOPE	$\mathrm{SemDescr}\left(\{\text{SWS-RCh}\}, \left[\left(\mathrm{Ontology}_{\mathrm{Design}}, \text{OWL-Sgeneral}\right)\right]\right)$ $\mathrm{CCh\_Property}_1 =$ $\left(\mathrm{Capability}, \text{OWL-S-capability}, \text{OWL-S-capability-me}\right)$	(22-18)
OWL-S-Geodata	$\mathrm{SemDescr}\left(\{\mathrm{RCh}\}, \begin{bmatrix}\left(\mathrm{Ontology}_{\mathrm{Design}}, \text{OWL-Sgeneral}\right) \sqcap \\ \left(\mathrm{Ontology}_{\mathrm{Domain}}, \mathrm{Geodata}\right)\end{bmatrix}\right)$	(22-19)
Reliability	$\mathrm{SemDescr}\left(\{\text{Nfp-RCh}\}, \begin{bmatrix}\left(\mathrm{Ontology}_{\mathrm{nfp}}, \mathrm{Service\_Reliability}\right) \sqcap \\ (\forall \mathit{Can\_be\_used\_by}.\mathrm{Administrator})\end{bmatrix}\right)$	(22-20)
WSMO	$\mathrm{SemDescr}\left(\{\text{SWS-RCh}\}, \left[\left(\mathrm{Ontology}_{\mathrm{Design}}, \mathrm{WSMOgeneral}\right)\right]\right)$	(22-21)
Travel	$\mathrm{SemDescr}(\{\mathrm{RCh}\}, [(\mathrm{Ontology}_{\mathrm{Domain}}, \mathrm{TravelBooking})])$ $\mathrm{CCh\_Property}_1 = \left(\mathrm{Journey}, \mathrm{journeySpec}, \text{journeySpec-me}\right)$ $\mathrm{CCh\_Property}_2 = \left(\mathrm{Cost}, \mathrm{costSpec}, \text{costSpec-me}\right)$	(22-22)
Insurance	$\mathrm{SemDescr}(\{\mathrm{RCh}\}, [(\mathrm{Ontology}_{\mathrm{Domain}}, \mathrm{TravelInsurance})])$ $\mathrm{CCh\_Property}_1 = \left(\mathrm{Policy}, \mathrm{policySpec}, \text{policySpec-me}\right)$	(22-23)

Table 34 - Request Characteristics (RCh)

B NFP-Ontologies

Based on the conceptual model provided in [O'EdHo2005] the WSMO Deliverable D28.4 v0.1 [TomFox2006] developed a set of ontologies and provided complete descriptions of these ontologies in WSML. The non-functional properties were modeled as ontologies in WSML listed in Table 35 Some of these ontologies are listed here in the appendix.

Nfp-ontologies in WSML	**Description**
Availability	The availability of a service combines temporal and locative aspects of the service to describe when and where one can interact with the service.
Currency	The Currency Ontology is a simple ontology that contains the most used currencies.
Discounts	Closely related to the notions of price and payment is the notion of Discount. Discounts are view from the service requestor perspective and are categorized according to the payment method and requestor's identity.
Intellectual Property	The Intellectual Property Ontology provides the concepts that are needed to describe Intellectual Property aspects. Main concepts include: IPRight, Trademark, Patent, Design, etc.
Location	The Locative Model is used to model the location of a service. Concepts like: Address, Region, Route, Point, Street Directory Reference, PhoneNumber, URI, IPAddress and Spectrum are directly related to this model.
Measures	The Measures Ontology provides a general measures terminology. Main concepts include: UnitOfMeasure, MeasurableQuantity, Distance, etc.
Obligation	The Obligations model captures the responsibilities of both service requestor and service provider. Three kinds of obligations were defined: Pricing obligations, Payment obligations and Relationship obligations.
Payment	The Payment model captures the manner in which a service requestor can fulfil their payment obligations. As stated before, payment and price are complementary.
Penalties	The Penalties are used by a service provider to specify what exactly will occur if a service requestor does not comply with a specific obligation. The same should hold the other way around. Penalties should be described by both the service provider and service requester and should apply to both.
Price	Price and Payment are seen as complementary non-functional properties. They represent two views of the same thing but from different perspectives. The payment (cost) is the user's perspective and the price is the provider's perspective.
Provider	The Service Provider model captures information about: the service identifier which can be a UNSPSC12 code, the service name and the provider of the service.

[12] The **United Nations Standard Products and Services Code** (**UNSPSC**) is a taxonomy of products and services for use in e-commerce. It is a five-level hierarchy coded as an 8-digit number.

Nfp-ontologies in WSML	Description
Quality-of-Service (QoS)	Quality is described relative to a standard, an industrial benchmark and/or a ranking schema.
Rewards	The Rewards Ontology includes concepts such as AccumulatedReward, AccumulatedPriceReward, RedeemableReward, etc
Rights	The Rights model captures the permissions granted to service providers and service requestors to perform operations.
Security	The Security model is attached to the locative aspect of the service and it's divided in two dimensions: identification and confidentiality.
Time	The Temporal Model provides the temporal concepts that are needed for time related descriptions of a service. These are: Temporal Date, Time, Temporal Interval and Temporal Duration. These concepts can further be refined in more specific concepts like Calendar Date for example.
Trust	Trust is a notion understood in various ways by different people. The Trust model is directly influenced by other models like endorsement and service inception.

Table 35 - Nfp-ontologies

B-1 NFP-Ontology for availability

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

namespace { _"http://www.wsmo.org/ontologies/nfp/availabilityNFPOntology#",
        dc _"http://purl.org/dc/elements/1.1#",
        xsd _"http://www.w3.org/2001/XMLSchema#",
        wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
        loc _"http://www.wsmo.org/ontologies/nfp/locativeNFPOntology#",
        temp _"http://www.wsmo.org/ontologies/nfp/temporalNFPOntology#",
        qua _"http://www.wsmo.org/ontologies/nfp/qualityNFPOntology#"
}

ontology _"http://www.wsmo.org/ontologies/nfp/availabilityNFPOntology"
        nonFunctionalProperties
                dc#title hasValue "Availability Ontology"
                dc#type hasValue _"http://www.wsmo.org/2004/d2#ontologies"
                dc#format hasValue "text/html"
                dc#identifier hasValue _"http://www.wsmo.org/ontologies/nfp/availabilityNFPOntology"
                dc#language hasValue "en-US"
                wsml#version hasValue "$Revision: 1.0 $"
        endNonFunctionalProperties

        concept Availability
                nonFunctionalProperties
                        dc#description hasValue "Availability in terms of when, where, and to whom something is available"
                endNonFunctionalProperties
                isAvailableAt ofType (1 *) loc#LocativeEntity
                isAvailableDuring ofType (1 *) temp#TemporalEntity
                isAvailableTo ofType (1 *) _iri

        concept RequestAvailability subConceptOf Availability
                nonFunctionalProperties
                        dc#description hasValue "links Availability to request"
                endNonFunctionalProperties
                forRequest ofType (1 1) _iri
                hasNegotiableTime ofType (0 1) _boolean
                isContinuouslyAvailable ofType (0 1) _boolean
```

B-2 NFP-Ontology for provider

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

namespace { _"http://www.wsmo.org/ontologies/nfp/providerNFPOntology#",
        dc _"http://purl.org/dc/elements/1.1#",
        xsd _"http://www.w3.org/2001/XMLSchema#",
        wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
        loc _"http://www.wsmo.org/ontologies/nfp/locativeNFPOntology#",
        tmp _"http://www.wsmo.org/ontologies/nfp/temporalNFPOntology#",
        obl _"http://www.wsmo.org/ontologies/nfp/obligationsNFPOntology#",
        qua _"http://www.wsmo.org/ontologies/nfp/qualityNFPOntology#"
}

ontology _"http://www.wsmo.org/ontologies/nfp/providerNFPOntology"
        nonFunctionalProperties
                dc#title hasValue "Provider Ontology"
                dc#creator hasValue {"Ioan Toma"}
                dc#subject hasValue {"Provider"}
                dc#description hasValue "Provider Ontology"
                dc#publisher hasValue "DERI Innsbruck"
                dc#contributor hasValue {"Ioan Toma"}
                dc#date hasValue "2006-05-08"
                dc#type hasValue _"http://www.wsmo.org/2004/d2#ontologies"
                dc#format hasValue "text/html"
             dc#identifier hasValue _"http://www.wsmo.org/ontologies/nfp/providerNFPOntology"
                dc#language hasValue "en-US"
                wsml#version hasValue "$Revision: 1.0 $"
        endNonFunctionalProperties

        concept Provider
                nonFunctionalProperties
                        dc#description hasValue "Provider concept definition"
                endNonFunctionalProperties
                hasName ofType (1 1) _string
                hasPriceMatching ofType (1 1) PriceMatching
                hasCompliance ofType AchievedCompliance
                offersPriceMatching ofType (1 1) PriceMatching
                hasProviderFeedback ofType (0 1) Endorsement
                hasMissionStatement ofType (0 1) Statement
                isLegallyBoundBy ofType (1 *) Legislation
                hasYearOfInception ofType (0 1) _year
                hasProviderMembership ofType ProviderMembership
                hasAssociationWith ofType AssociationTypeProvider

        concept AssociationTypeProvider
                nonFunctionalProperties
                        dc#description hasValue "AssociationTypeProvider concept definition"
                endNonFunctionalProperties
                hasAssociationType ofType (1 1) AssociationType
                withProvider ofType (1 1) Provider

        concept ProviderMembership
                nonFunctionalProperties
                        dc#description hasValue "ProviderMembership concept definition"
                endNonFunctionalProperties
                providerOf ofType (1 1) Provider
                hasMembership ofType (1 1) obl#Membership
                wasAchievedOn ofType (1 1) tmp#TemporalDate
                hasMembershipExpiryOf ofType (1 1) tmp#TemporalDate

        concept AchievedCompliance
                nonFunctionalProperties
                        dc#description hasValue "AchievedCompliance concept definition"
                endNonFunctionalProperties
                hasCompliance ofType (1 1) Compliance
                hasConformanceRating ofType (1 1) StandardLevelName
                wasAchievedOn ofType (1 1) tmp#TemporalDate
                wasVerifiedBy ofType (1 1) Provider

        concept Compliance
                nonFunctionalProperties
                        dc#description hasValue "Compliance concept definition"
                endNonFunctionalProperties
                achievedConformanceOfStandard ofType (1 1) qua#Standard
                forService ofType (1 1) _iri //Service
        concept AssociationType
                nonFunctionalProperties
                        dc#description hasValue "AssociationType concept definition"
                endNonFunctionalProperties

        concept PartnerType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "PartnerType concept definition"
                endNonFunctionalProperties

        concept SubsidiaryType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "SubsidiaryType concept definition"
                endNonFunctionalProperties

        concept OwnerType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "OwnerType concept definition"
                endNonFunctionalProperties

        concept SupplierToType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "SupplierToType concept definition"
                endNonFunctionalProperties

        concept AgencyType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "AgencyType concept definition"
                endNonFunctionalProperties

        concept DivisionType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "DivisionType concept definition"
                endNonFunctionalProperties

        concept BranchType subConceptOf AssociationType
                nonFunctionalProperties
                        dc#description hasValue "BranchType concept definition"
                endNonFunctionalProperties
```

B-3 NFP-Ontology for location (locative)

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

        namespace { _"http://www.wsmo.org/ontologies/nfp/locativeNFPOntology#",
                dc _"http://purl.org/dc/elements/1.1#",
                xsd _"http://www.w3.org/2001/XMLSchema#",
                wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
                temp _"http://www.wsmo.org/ontologies/nfp/temporalNFPOntology",
                qu _"http://www.wsmo.org/ontologies/nfp/qualityNFPOntology",
                sec _"http://www.wsmo.org/ontologies/nfp/qualityNFPOntology",
                meas _"http://www.wsmo.org/ontologies/nfp/measuresNFPOntology"
}

ontology _"http://www.wsmo.org/ontologies/nfp/locativeNFPOntology"
        nonFunctionalProperties
                dc#title hasValue "Locative Ontology"
        endNonFunctionalProperties

importsOntology {_"http://www.wsmo.org/ontologies/nfp/temporalNFPOntology",
        _"http://www.wsmo.org/ontologies/nfp/qualityNFPOntology",
        _"http://www.wsmo.org/ontologies/nfp/securityNFPOntology",
        _"http://www.wsmo.org/ontologies/nfp/measuresNFPOntology"}

        concept LocativeEntity
                nonFunctionalProperties
                        dc#description hasValue "LocativeEntity concept definition"
                endNonFunctionalProperties
                hasName ofType _string
                supportsWrittenLanguage ofType Language
                supportsSpokenLanguage ofType Language
                canBeComunicatedAccordingTo ofType qu#Standard
                hasIdentificationRequirement ofType (1) sec#IdentificationRequirement
                hasConfidentiality ofType (1) sec#Confidentiality
```

```
concept GeoLocation subConceptOf LocativeEntity
        nonFunctionalProperties
                dc#description hasValue "GeoLocation concept definition"
                dc#relation hasValue { validLatitude, validLongitude, validPoleGeoLocation}
        endNonFunctionalProperties
        hasLatitude ofType (1) meas#Angle
        hasLongitude ofType (1 1) meas#Angle
        hasAltitude ofType (0 1) meas#Distance

axiom validLatitude
        definedBy
        !- ?point [latitude hasValue ?LA] memberOf GeoLocation and
        ?LA [numUnits hasValue ?NU, ofUnits hasValue meas#DegreeOfArc] memberOf meas#Angle and
                (less(90, ?NU) or greater(_decimal("-90"), ?NU)).

axiom validLongitude
        definedBy
                !- ?point [longitude hasValue ?LO] memberOf GeoLocation and
                ?LO [numUnits hasValue ?NU, ofUnits hasValue meas#DegreeOfArc]  memberOf meas#Angle and
                (less(180, ?NU) or greaterEqual(_decimal("-180"), ?NU)).

axiom validPoleGeoLocation
        definedBy
                !- ?point [latitude hasValue ?LA, longitude hasValue ?LO] memberOf GeoLocation and
                ?LA [numUnits hasValue 90, ofUnits hasValue meas#DegreeOfArc] memberOf meas#Angle and
                ?LO [numUnits hasValue ?NU, ofUnits hasValue meas#DegreeOfArc]  memberOf meas#Angle.

concept CompassDirection
        nonFunctionalProperties
                dc#description hasValue "Direction concept definition"
                dc#relation hasValue validCompassDirection
        endNonFunctionalProperties
        ofUnits ofType meas#UnitOfArc

axiom validCompassDirection
        definedBy
                !- ?dir [numUnits hasValue ?ANGLE, ofUnits hasValue meas#DegreeOfArc]
                memberOf CompassDirection and (lessEqual(360, ?ANGLE) or greater(0, ?ANGLE)).

concept VehicularRoute subConceptOf LocativeEntity
        nonFunctionalProperties
                dc#description hasValue "VehicularRoute concept definition"
        endNonFunctionalProperties
        hasName ofType (1 1) _string
        hasSpecification ofType RouteSpecification
        intendedForVehicles ofType (1 *) Vehicle
        possiblePathFor ofType (1 *) Vehicle

concept VehicleType
        nonFunctionalProperties
                dc#description hasValue "VehicleType"
        endNonFunctionalProperties

concept PathThroughWater subConceptOf VehicularRoute
        nonFunctionalProperties
                dc#description hasValue "PathThroughWater concept definition"
        endNonFunctionalProperties

concept WaterSurfacePath subConceptOf PathThroughWater
        nonFunctionalProperties
                dc#description hasValue "WaterSurfacePath concept definition"
        endNonFunctionalProperties

concept Canal subConceptOf WaterSurfacePath
        nonFunctionalProperties
                dc#description hasValue "Canal concept definition"
        endNonFunctionalProperties

concept SeaLane subConceptOf WaterSurfacePath
        nonFunctionalProperties
                dc#description hasValue "SeaLane concept definition"
        endNonFunctionalProperties

concept SolidSurfacePathThroughAir subConceptOf VehicularRoute
        nonFunctionalProperties
                dc#description hasValue "SolidSurfacePathThroughAir concept definition"
        endNonFunctionalProperties
```

```
concept PathForWheeledVehicles subConceptOf SolidSurfacePathThroughAir
        nonFunctionalProperties
                dc#description hasValue "PathForWheeledVehicles concept definition"
        endNonFunctionalProperties

concept Railway subConceptOf PathForWheeledVehicles
        nonFunctionalProperties
                dc#description hasValue "Railway concept definition"
        endNonFunctionalProperties

concept Subway subConceptOf Railway
        nonFunctionalProperties
                dc#description hasValue "Subway concept definition"
        endNonFunctionalProperties

concept Roadway subConceptOf PathForWheeledVehicles
        nonFunctionalProperties
                dc#description hasValue "Roadway concept definition"
        endNonFunctionalProperties

concept RoadLane subConceptOf PathForWheeledVehicles
        nonFunctionalProperties
                dc#description hasValue "RoadLane concept definition"
        endNonFunctionalProperties

concept Driveway subConceptOf PathForWheeledVehicles
        nonFunctionalProperties
                dc#description hasValue "Driveway concept definition"
        endNonFunctionalProperties

concept BicyclePath subConceptOf PathForWheeledVehicles
        nonFunctionalProperties
                dc#description hasValue "BicyclePath concept definition"
        endNonFunctionalProperties

concept SkiSlope subConceptOf SolidSurfacePathThroughAir
        nonFunctionalProperties
                dc#description hasValue "SkiSlope concept definition"
        endNonFunctionalProperties

concept SkiJump subConceptOf SkiSlope
        nonFunctionalProperties
                dc#description hasValue "SkiJump concept definition"
        endNonFunctionalProperties

concept Footpath subConceptOf SolidSurfacePathThroughAir
        nonFunctionalProperties
                dc#description hasValue "Footpath concept definition"
        endNonFunctionalProperties

concept Stairway subConceptOf Footpath
        nonFunctionalProperties
                dc#description hasValue "Stairway concept definition"
        endNonFunctionalProperties

concept MovingWalkway subConceptOf Footpath
        nonFunctionalProperties
                dc#description hasValue "MovingWalkway concept definition"
        endNonFunctionalProperties

concept MovingStairway subConceptOf MovingWalkway
        nonFunctionalProperties
                dc#description hasValue "MovingStairway concept definition"
        endNonFunctionalProperties

concept Trail subConceptOf Footpath
        nonFunctionalProperties
                dc#description hasValue "Trail concept definition"
        endNonFunctionalProperties

concept Sidewalk subConceptOf Footpath
        nonFunctionalProperties
                dc#description hasValue "Sidewalk concept definition"
        endNonFunctionalProperties

concept GangPlank subConceptOf Footpath
        nonFunctionalProperties
```

```
                dc#description hasValue "GangPlank concept definition"
        endNonFunctionalProperties

concept PathThroughAir subConceptOf VehicularRoute
        nonFunctionalProperties
                dc#description hasValue "PathThroughAir concept definition"
        endNonFunctionalProperties

concept AirLane subConceptOf PathThroughAir
        nonFunctionalProperties
                dc#description hasValue "AirLane concept definition"
        endNonFunctionalProperties

concept GlideSlope subConceptOf PathThroughAir
        nonFunctionalProperties
                dc#description hasValue "GlideSlope concept definition"
        endNonFunctionalProperties

concept IndicativeRouteType
        nonFunctionalProperties
                dc#description hasValue "IndicativeRouteType concept definition"
        endNonFunctionalProperties
        value ofType _string

concept RouteSpecification
        nonFunctionalProperties
                dc#description hasValue "RouteSpecification concept definition"
        endNonFunctionalProperties
        hasSpecification ofType (1 *) RouteSpecificationType

concept RouteSpecificationType
        nonFunctionalProperties
                dc#description hasValue "RouteSpecificationType concept definition"
        endNonFunctionalProperties
        hasPoint ofType (1 *) GeoLocation
        hasOrder ofType (1 1) nonNegativeInteger

concept RouteSpecification
        nonFunctionalProperties
                dc#description hasValue "RouteSpecification concept definition"
        endNonFunctionalProperties
        hasNthRoutePoint ofType (1 *) NthRoutePoint

concept NthRoutePoint
        nonFunctionalProperties
                dc#description hasValue "NthRoutePoint concept definition"
        endNonFunctionalProperties
        hasPoint ofType (1) GeoLocation
        hsOrder ofType (1) nonNegativeInteger

concept Region subConceptOf LocativeEntity
        nonFunctionalProperties
                dc#description hasValue "Region concept definition"
        endNonFunctionalProperties
        hasName ofType (1) _string
        hasSpecification ofType RegionSpecification

concept GeopoliticalPlace subConceptOf LocativeEntity
        nonFunctionalProperties
                dc#description hasValue "GeopoliticalPlace concept definition"
        endNonFunctionalProperties

concept GeoLocation subConceptOf GeopoliticalPlace
        nonFunctionalProperties
                dc#description hasValue "GeoLocation concept definition"
        endNonFunctionalProperties

concept GeographicalRegion subConceptOf {GeographicalPlace, Region}
        nonFunctionalProperties
                dc#description hasValue "GeographicalRegion concept definition"
        endNonFunctionalProperties

concept GeopoliticalRegion subConceptOf GeographicalRegion
nonFunctionalProperties
                dc#description hasValue "GeopoliticalRegion concept definition"
        endNonFunctionalProperties
```

```
concept HumanResidenceArea subConceptOf GeographicalRegion
nonFunctionalProperties
                dc#description hasValue "HumanResidenceArea concept definition"
        endNonFunctionalProperties

concept UrbanArea subConceptOf HumanResidenceArea
nonFunctionalProperties
                dc#description hasValue "UrbanArea concept definition"
        endNonFunctionalProperties

concept Neighborhood subConceptOf HumanResidenceArea
nonFunctionalProperties
                dc#description hasValue "Neighborhood concept definition"
        endNonFunctionalProperties

concept SuburbanArea subConceptOf HumanResidenceArea
nonFunctionalProperties
                dc#description hasValue "SuburbanArea concept definition"
        endNonFunctionalProperties

concept NationalTerritory subConceptOf GeopoliticalRegion
        nonFunctionalProperties
                dc#description hasValue "NationalTerritory concept definition"
        endNonFunctionalProperties

concept SubnationalTerritory subConceptOf GeopoliticalRegion
        nonFunctionalProperties
                dc#description hasValue "SubnationalTerritory concept definition"
        endNonFunctionalProperties

concept CityTerritory subConceptOf SubnationalTerritory
        nonFunctionalProperties
                dc#description hasValue "CityTerritory concept definition"
        endNonFunctionalProperties

concept CapitolTerritory subConceptOf CityTerritory
        nonFunctionalProperties
                dc#description hasValue "CapitolTerritory concept definition"
        endNonFunctionalProperties

concept PrimarySubnationalTerritory subConceptOf SubnationalTerritory
        nonFunctionalProperties
                dc#description hasValue "PrimarySubnationalTerritory concept definition; e.g. State, Provice"

        endNonFunctionalProperties

concept SecondarySubnationalTerritory subConceptOf SubnationalTerritory
        nonFunctionalProperties
                dc#description hasValue "SecondarySubnationalTerritory concept definition; e.g. Country, Parish"

        endNonFunctionalProperties

concept PostalCodeArea subConceptOf SubnationalTerritory
        nonFunctionalProperties
                dc#description hasValue "PostalCodeArea concept definition"
        endNonFunctionalProperties

concept SchoolDistrictTerritory subConceptOf SubnationalTerritory
        nonFunctionalProperties
                dc#description hasValue "SchoolDistrictTerritory concept definition"
        endNonFunctionalProperties

concept ControlledLand subConceptOf GeopoliticalRegion
        nonFunctionalProperties
                dc#description hasValue "ControlledLand concept definition"
        endNonFunctionalProperties

concept ColonialTerritory subConceptOf ControlledLand
        nonFunctionalProperties
                dc#description hasValue "ColonialTerritory concept definition"
        endNonFunctionalProperties

concept DominionTerritory subConceptOf ControlledLand
        nonFunctionalProperties
                dc#description hasValue "DominionTerritory concept definition"
        endNonFunctionalProperties
```

```
concept OccupiedTerritory subConceptOf ControlledLand
        nonFunctionalProperties
                dc#description hasValue "OccupiedTerritory concept definition"
        endNonFunctionalProperties

concept BodyOfLand subConceptOf GeographicalRegion
        nonFunctionalProperties
                dc#description hasValue "BodyOfLand concept definition"
        endNonFunctionalProperties

concept Continent subConceptOf BodyOfLand
        nonFunctionalProperties
                dc#description hasValue "Continent concept definition"
        endNonFunctionalProperties

concept Subcontinent subConceptOf BodyOfLand
        nonFunctionalProperties
                dc#description hasValue "Subcontinent concept definition"
        endNonFunctionalProperties

concept Island subConceptOf BodyOfLand
        nonFunctionalProperties
                dc#description hasValue "Island concept definition"
        endNonFunctionalProperties

concept Archipelago subConceptOf BodyOfLand
        nonFunctionalProperties
                dc#description hasValue "Archipelago concept definition"
        endNonFunctionalProperties

concept LandTopographicalFeature subConceptOf GeographicalRegion
        nonFunctionalProperties
                dc#description hasValue "LandTopographicalFeature concept definition"
        endNonFunctionalProperties

concept Peninsula subConceptOf LandTopographicalFeature
        nonFunctionalProperties
                dc#description hasValue "Peninsula concept definition"
        endNonFunctionalProperties

concept Isthmus subConceptOf LandTopographicalFeature
        nonFunctionalProperties
                dc#description hasValue "Isthmus concept definition"
        endNonFunctionalProperties

concept Plateau subConceptOf LandTopographicalFeature
        nonFunctionalProperties
                dc#description hasValue "Plateau concept definition"
        endNonFunctionalProperties

concept MountainRange subConceptOf LandTopographicalFeature
        nonFunctionalProperties
                dc#description hasValue "MountainRange concept definition"
        endNonFunctionalProperties


concept Desert subConceptOf LandEcologicalRegion
        nonFunctionalProperties
                dc#description hasValue "Desert concept definition"
        endNonFunctionalProperties

concept Glacier subConceptOf LandEcologicalRegion
        nonFunctionalProperties
                dc#description hasValue "Glacier concept definition"
        endNonFunctionalProperties

concept Wetland subConceptOf LandEcologicalRegion
        nonFunctionalProperties
                dc#description hasValue "Wetland concept definition"
        endNonFunctionalProperties

concept LandEcologicalRegion subConceptOf GeographicalRegion
        nonFunctionalProperties
                dc#description hasValue "LandEcologicalRegion concept definition"
        endNonFunctionalProperties

concept Forest subConceptOf LandEcologicalFeature
```

```
                nonFunctionalProperties
                        dc#description hasValue "Forest concept definition"
                endNonFunctionalProperties

        concept RainForest subConceptOf Forest
                nonFunctionalProperties
                        dc#description hasValue "RainForest concept definition"
                endNonFunctionalProperties

        concept Savannah subConceptOf LandEcologicalFeature
                nonFunctionalProperties
                        dc#description hasValue "Savannah concept definition"
                endNonFunctionalProperties

        concept Steppe subConceptOf LandEcologicalFeature
                nonFunctionalProperties
                        dc#description hasValue "Steppe concept definition"
                endNonFunctionalProperties

        concept BodyOfWater subConceptOf GeographicalRegion
                nonFunctionalProperties
                        dc#description hasValue "BodyOfWater concept definition"
                endNonFunctionalProperties

        concept Ocean subConceptOf BodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "Ocean concept definition"
                endNonFunctionalProperties

        concept Lake subConceptOf BodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "Lake concept definition"
                endNonFunctionalProperties

        concept WaterStream subConceptOf BodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "WaterStream concept definition"
                endNonFunctionalProperties

        concept River subConceptOf WaterStream
                nonFunctionalProperties
                        dc#description hasValue "River concept definition"
                endNonFunctionalProperties

        concept Creek subConceptOf WaterStream
                nonFunctionalProperties
                        dc#description hasValue "Creek concept definition"
                endNonFunctionalProperties

        concept PartialBodyOfWater subConceptOf BodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "PartialBodyOfWater concept definition"
                endNonFunctionalProperties

        concept BayGulf subConceptOf PartialBodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "BayGulf concept definition"
                endNonFunctionalProperties

        concept Sea subConceptOf PartialBodyOfWater
                nonFunctionalProperties
                        dc#description hasValue "Sea concept definition"
                endNonFunctionalProperties

concept RegionSpecification
                nonFunctionalProperties
                        dc#description hasValue "RegionSpecification concept definition"
                endNonFunctionalProperties
                hasSpecification ofType (3 *) NthBorderPoint
                numberOfBorderPoints ofType nonNegativeInteger

        concept NthBorderPoint
                nonFunctionalProperties
                        dc#description hasValue "NthBorderPoint concept definition"
                endNonFunctionalProperties
          hasPoint ofType (1) GeoLocation
          hasOrder ofType (1) _integer
        concept Address subConceptOf LocativeEntity
        nonFunctionalProperties
        dc#description hasValue "Address concept definition"
        endNonFunctionalProperties
  hasCountry ofType (0 1) Country
  hasCountrySubdivision ofType (0 1) PrimarySubnationalTerritory
  hasCountrySubSubdivision ofType (0 1) SecondarySubnationalTerritory
  hasTown ofType (0 1) CityTerritory //includes village, town, ...
  hasSubTown ofType (0 *) GeopoliticalRegion
  hasTeritory ofType (0 1) ControlledLand
  hasPostcode ofType (0 1) _string
  hasAddressee ofType (0 1) Addressee
  inSupraNationalRegion ofType (0 1) GeographicalRegion

concept PostBoxAddress subConceptOf Address
        nonFunctionalProperties
        dc#description hasValue "Postbox Address "
        endNonFunctionalProperties
  hasPostBoxNumber ofType _string

        concept StreetAddress subConceptOf {Address, GeographicalRegion}
        nonFunctionalProperties
        dc#description hasValue "Street Address"
        endNonFunctionalProperties
  hasStreetType ofType _string
  hasStreetName ofType _string
        hasStreetNumber ofType _string
        hasStreetDirectoryReference ofType StreetDirectoryReference
        hasProximity ofType Proximity
        carrierInstructions ofType _string

concept InternalAddress subConceptOf StreetAddress
        nonFunctionalProperties
        dc#description hasValue "Address internal to a StreetAddress"
        endNonFunctionalProperties
        hasBuildingID ofType _string
        hasLevel ofType _string
        hasUnitID ofType _string
        hasRoomNumber ofType _string
        mailStop ofType _string
  internalRoutingInstructions ofType _string

        concept PostBoxType
                nonFunctionalProperties
                        dc#description hasValue "PostBoxType concept definition"
                endNonFunctionalProperties
                value ofType _string

relation spatiallyRelated ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean)

relation near ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation adjacentTo ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf near

relation touching ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf near

relation inGeneric ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation inPartially ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inGeneric

relation inAmong ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inGeneric

relation inSurrounded ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inGeneric

relation inEmbedded ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inSurrounded

relation subRegions ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inGeneric

relation borderSubRegions ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf subRegions

relation internalSubRegions ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf {subRegions,
        inSurrounded}

relation internalParts ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf inSurrounded

relation aboveGeneric ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated
relation aboveGenerally ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf aboveGeneric

relation aboveHigerThan ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf aboveGeneric

relation northOf ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation southOf ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation eastOf ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation westOf ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation facing ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

relation levelWith ( ofType LocativeEntity, ofType LocativeEntity, impliesType _boolean) subRelationOf spatiallyRelated

concept Addressee
            nonFunctionalProperties
                  dc#description hasValue "Addressee concept definition"
            endNonFunctionalProperties
      hasDepartmentName ofType (0 1) _string
      hasName ofType (0 1) _string
      hasFunctionalTitle ofType (0 1) _string
      hasProfessionalTitle ofType (0 1) _string
      hasOrganizationName ofType (0 1) _string

concept PhoneNumber subConceptOf LocativeEntity
            nonFunctionalProperties
                  dc#description hasValue "PhoneNumber concept definition"
            endNonFunctionalProperties
      hasType ofType (1 1) PhoneLineType
      hasInteractionType ofType (1 1) PhoneNumberInteractionType
      tollFreeCallForCallersFromRegion ofType (0 1) Region
      hasCountryCode ofType (0 1) nonNegativeInteger
      hasNationalDirectDialPrefix ofType (0 1) nonNegativeInteger
      hasCityOrAreaCode ofType (0 1) nonNegativeInteger
      hasLocalNumber ofType (1 1) nonNegativeInteger
      hasInternationalPrefix ofType InternationalPrefixForRegion

      concept PhoneLineType
            nonFunctionalProperties
                  dc#description hasValue "PhoneLineType concept definition"
            endNonFunctionalProperties

      instance MobileNumberType memberOf PhoneLineType
      instance CellNumberType memberOf PhoneLineType
      instance FixedLineNumberType memberOf PhoneLineType

      concept InternationalPrefixForRegion
            nonFunctionalProperties
                  dc#description hasValue "InternationalPrefixForRegion concept definition"
            endNonFunctionalProperties
            hasInternationalDirectDialPrefix ofType (1 1) nonNegativeInteger
            forCallersFromRegion ofType (1 *) Region

concept IPAddress subConceptOf LocativeEntity
            nonFunctionalProperties
                  dc#description hasValue "IPAddress concept definition"
            endNonFunctionalProperties
            //todo - to be defined

concept EthernetAddress subConceptOf LocativeEntity
            nonFunctionalProperties
                  dc#description hasValue "EthernetAddress concept definition"
            endNonFunctionalProperties
            //todo - to be defined

concept StreetDirectory
            nonFunctionalProperties
                  dc#description hasValue "StreetDirectory concept definition"
            endNonFunctionalProperties
      hasEdition ofType (0 1) nonNegativeInteger
      hasProvider ofType (0 1) Provider
      hasISBNCode ofType (0 1) _string
      hasPublicationTitle ofType (1 1) _string
            hasPublicationDate ofType (0 1) temp#TemporalDate
```

```
  concept StreetDirectoryReference subConceptOf LocativeEntity
                    nonFunctionalProperties
                            dc#description hasValue "StreetDirectoryReference concept definition"
                    endNonFunctionalProperties
            hasXPosition ofType (1 1) _string
            hasYPosition ofType (1 1) _string
            hasRegion ofType (0 *) Region
            hasMapNumber ofType (1 1) _string
                    hasReference ofType (1 *) StreetDirectory

            concept nonNegativeInteger subConceptOf _integer
                    nonFunctionalProperties
                            dc#description hasValue "Non negative integer"
                            dc#relation hasValue validNonNegativeInteger
                    endNonFunctionalProperties

            axiom validNonNegativeInteger
                    definedBy
                            !- ?x memberOf _integer and lessThan(?x, 0).
```

B-4 NFP-Ontology for discount

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"

namespace { _"http://www.wsmo.org/ontologies/nfp/discountsNFPOntology#",
        dc _"http://purl.org/dc/elements/1.1#",
        xsd _"http://www.w3.org/2001/XMLSchema#",
        wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
        ava _"http://www.wsmo.org/ontologies/nfp/availabilityNFPOntology#",
        price _"http://www.wsmo.org/ontologies/nfp/priceNFPOntology#",
        pay _"http://www.wsmo.org/ontologies/nfp/paymentNFPOntology#",
        loc _"http://www.wsmo.org/ontologies/nfp/locativeNFPOntology#",
        temp _"http://www.wsmo.org/ontologies/nfp/temporalNFPOntology#"
}

ontology _"http://www.wsmo.org/ontologies/nfp/discountsNFPOntology"
        nonFunctionalProperties
                dc#title hasValue "Discounts Ontology"
        endNoFunctionalProperties

        concept Discount
                nonFunctionalProperties
                        dc#description hasValue "Discount concept definition"
                endNonFunctionalProperties
                hasCondition ofType Condition
                hasAmount ofType DiscountAmount
                hasResultingDiscountedPrice ofType ResultingDiscountPrice
                hasAvailability ofType (1 *) ava#Availability

        concept PayeeDiscount subConceptOf Discount
                nonFunctionalProperties
                        dc#description hasValue "PayeeDiscount concept definition"
                endNonFunctionalProperties

        concept StudentDiscount subConceptOf PayeeDiscount
                nonFunctionalProperties
                        dc#description hasValue "StudentDiscount concept definition"
                endNonFunctionalProperties
                applicableToSchoolStudents ofType (1 1) _boolean
                applicableToFulltimeUniversityStudents ofType (1 1) _boolean

        concept MembershipDiscount subConceptOf PayeeDiscount
                nonFunctionalProperties
                        dc#description hasValue "MembershipDiscount concept definition"
                endNonFunctionalProperties
                isAvailableToHolders ofType Membership

        concept ShareholderDiscount subConceptOf PayeeDiscount
                nonFunctionalProperties
                        dc#description hasValue "ShareholderDiscount concept definition"
                endNoFunctionalProperties
                availableToShareholders ofType _iri //Provider
                availableToShareholdersWithMinimumNumberOfUnits ofType (1 1) loc#nonNegativeInteger

        concept AgeGroupDiscount subConceptOf PayeeDiscount
        nonFunctionalProperties
                dc#description hasValue "AgeGroupDiscount concept definition"
        endNonFunctionalProperties
        hasName ofType (0 1) _string
        ageFromValue ofType (1 1) loc#nonNegativeInteger
        ageToValue ofType (1 1) loc#nonNegativeInteger

concept DiscountAmount
        nonFunctionalProperties
                dc#description hasValue "DiscountAmount concept definition"
        endNonFunctionalProperties
        absoluteDiscount ofType (0 1) price#MonetaryAmount
        percetDiscount ofType (0 1) price#Percentage
        forService ofType (1 1) _iri //Service

concept ResultingDiscountedPrice subConceptOf price#Price
        nonFunctionalProperties
                dc#description hasValue "ResultingDiscountedPrice concept definition"
        endNonFunctionalProperties

concept PaymentDiscount
        nonFunctionalProperties
                dc#description hasValue "PaymentDiscount concept definition"
        endNonFunctionalProperties
        hasMinimumPriceRequiredToReceiveDiscount ofType (0 1) price#AbsoutePrice

concept PaymentInstrumentTypeDiscount subConceptOf PaymentDiscount
        nonFunctionalProperties
                dc#description hasValue "PaymentInstrumentTypeDiscount concept definition"
        endNonFunctionalProperties
        offersPaymentInstrumentTypeDiscountFor ofType (1 1) pay#PaymentInstrumentType

concept PaymentLocationTypeDiscount subConceptOf PaymentDiscount
        nonFunctionalProperties
                dc#description hasValue "PaymentLocationTypeDiscount concept definition"
        endNonFunctionalProperties
        offersPaymentLocationTypeDiscountFor ofType (0 1) loc#LocativeEntityType
concept CouponPaymentDiscount subConceptOf PaymentDiscount
        nonFunctionalProperties
                dc#description hasValue "CouponPaymentDiscount concept definition"
        endNonFunctionalProperties
        hasValidityPeriod ofType (0 1) temp#TemporalEntity
        isIssuedBy ofType (1 1) _iri //could be a person, organization or any kind of provider

concept EarlyPaymentDiscount subConceptOf PaymentDiscount
        nonFunctionalProperties
                dc#description hasValue "CouponPaymentDiscount concept definition"
        endNonFunctionalProperties
        hasEarlyPaymentOffset ofType temp#TemporalDuration
        cutOffDate ofType (0 1) temp#TemporalDuration
```

BIBLIOGRAPHY

[**AkkFar et al.2005]** AKKIRAJU, RAMA; FARRELL, JOEL; MILLER, JOHN; NAGARAJAN, MEENAKSHI; SCHMIDT, MARC-THOMAS; SHETH, AMIT; VERMA, *Kunal.Web Service Semantics - WSDL-S: W3C Member Submission,* W3C 07.11.2005 [http://www.w3.org/Submission/WSDL-S/] (accessed 14. Apr. 2010).

[AkkGoo et al.2003] AKKIRAJU, R.; GOODWIN, R.; DOSHI, P.; ROEDER, S.; KAMBHAMPATI, S.; KNOBLOCK, C. A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. IIWeb. *Proceedings of IJCAI-03 Workshop on Information Integration on* **2003.**

[AloCas et al.2004] ALONSO, G.; CASATI, F.; KUNO, H.; MACHIRAJU, *V. Web Services: Concepts, Architectures and Applications.* Data-Centric Systems and Applications; Springer: Berlin, Heidelberg, 2004 (ISBN 3-540-44008-9).

[ama2009] *amazon web services,* Overview of Amazon Web Services; White Paper, amazon, December 2009 [http://awsmedia.s3.amazonaws.com/AWS_Overview_Whitepaper_120809.pdf].

[AmiWol2005] AMIN, M.S.; WOLLENBERG, B.F. Toward a smart grid: power delivery for the 21st century. Power and Energy Magazine, IEEE. *Power and Energy Magazine, IEEE* **2005**, *3* (5) pp. 34–41 [http://ieeexplore.ieee.org/ielx5/8014/32291/01507024.pdf?tp=&arnumber=1507024&isnumber=32291].

[AndCur et al.2003] ANDREWS, TONY; CURBERA, FRANCISCO; DHOLAKIA, HITESH; GOLAND, YARON; KLEIN, JOHANNES; LEYMANN, FRANK; LIU, KEVIN; ROLLER, DIETER; SMITH, DOUG; THATTE, SATISH; TRICKOVIC, IVANA; WEERAWARANA, *Sanjiva.BPEL V1-1: Business Process Execution Language (BPEL4WS),* BEA Systems; International Business Machines Corporation; Microsoft Corporation; SAP AG; Siebel Systems 06.05.2003 [http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf] (accessed 02. Apr. 2009).

[AnSaRa2008] ANTONIO BROGI; SARA CORFINI; RAZVAN POPESCU. Semantics-based composition-oriented discovery of Web services. *ACM Trans. Internet Technol.* **2008**, *8* (4) pp. 1–39 [doi:10.1145/1391949.1391953].

[ArdPro et al.2013] ARDITO, L.; PROCACCIANTI, G.; MENGA, G.; MORISIO, M. Smart Grid Technologies in Europe: An Overview. *Energies* **2013**, *6* (1) pp. 251–281 [http://www.mdpi.com/1996-1073/6/1/251].

[ArkAsk et al.2002] ARKIN, ASSAF; ASKARY, SID; JEKELI, WOLFGANG; KAWAGUCHI, KOHSUKE; FORDIN, *Scott.Web Service Choreography Interface (WSCI) 1.0*, W3C 09.08.2002 [http://www.w3.org/TR/wsci/] (accessed 25. Nov. 2010).

[Ars2006] ARS TECHNICA.*Tim Berners-Lee on Web 2.0: "nobody even knows what it means"*, Ars Technica 2006 [http://arstechnica.com/business/news/2006/09/7650.ars] (accessed 29. Mar. 2009).

[BahBur et al.1999] BAHR, K.; BURKHARDT, H.-J.; HOVESTADT, L.; REINEMA, R. Integrating Virtual and Real Work Environments. In *Proceedings of IEEE Conference SoftCOM'99*: *Software in Telecommunications and Computer Networks*. Split, Rijeka, Croatia & Triest, Venice, Italy, 1999.

[BanBar et al.2002] BANERJI, ARINDAM; BARTOLINI, CLAUDIO; BERINGER, DOROTHEA; CHOPELLA, VENKATESH; GOVINDARAJAN, KANNAN; KARP, ALAN; KUNO, HARUMI; LEMON, MIKE; POGOSSIANTS, GREGORY; SHARMA, SHAMIK; WILLIAMS, *Scott.Web Services Conversation Language (WSCL) 1.0: W3C Note 14 March 2002*, W3C; Hewlett-Packard Company 14.03.2002 [http://www.w3.org/TR/wscl10/] (accessed 30. Aug. 2010).

[Bar2010] BARTONITZ, *Martin Dr.BPM_Round-Trip-Engineering: Vision und Wirklichkeit; Gesellschaft für Informatik, Regionalgruppe Düsseldorf, 13.01.2010*, Saperion AG 2010 [http://www1.gi-ev.de/regionalgruppen/duesseldorf/] (accessed 07. Mar. 2010).

[BatBer et al.2005] BATTLE, STEVE; BERNSTEIN, ABRAHAM; BOLEY, HAROLD; GROSOF, BENJAMIN; GRUNINGER, MICHAEL; HULL, RICHARD; KIFER, MICHAEL; MARTIN, DAVID; MCILRAITH, SHEILA; MCGUINNESS, DEBORAH; SU, JIANWEN; TABET, *Said.Semantic Web Services Framework (SWSF) Overview: W3C Member Submission; 9 September 2005*, W3C 15.09.2005 [http://www.w3.org/Submission/SWSF/] (accessed 22. Jan. 2010).

[Bea1997] BEARMAN, *Mirion.Tutorial on ODP Trading Function. 14.03.1997 [*http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.6195] (accessed 26. Aug. 2010).

[BecHar et al.2004] BECHHOFER, SEAN; HARMELEN, FRANK VAN; HENDLER, JIM; HORROCKS, IAN; MCGUINNESS, DEBORAH L.; PATEL-SCHNEIDER, PETER F.; STEIN, *Lynn Andrea.OWL Web Ontology Language: Reference, W3C Recommendation 10 February 2004*, W3C 2004 [http://www.w3.org/TR/2004/REC-owl-ref-20040210/] (accessed 30. Mar. 2009).

[BeeEya et al.2006] BEERI, C.; EYAL, A.; KAMENKOVICH, S.; MILO, T. Querying business processes. In *Proceedings of the 32nd international conference on Very large data bases*, pp 343–354. VLDB Endowment: Seoul, Korea, 2006.

[BeKlMe2000] BEIERLE, C.; KLOOS, R.; MEYER, *G. A Pragmatic Type Concept for Prolog Supporting Polymorphism, Subtyping, and Meta-Programming.* In *ICLP'99 - International Conference on Logic Programming*: *Workshop on Verification of Logic Programs.* Las Cruces, New Mexico, USA. Electronic Notes in Theoretical Computer Science 30; Elsevier; MIT: Cambridge, Mass., 2000.

[BePoRi1999] BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. JADE - A FIPA-compliant agent framework. In *4th Intl. Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents,* pp 97–108. London, UK, 1999.

[Ber2003] BERNERS-LEE, *Tim.Web Services: Progam Integration across Application and Organization boundaries*, W3C 2003 [http://www.w3.org/DesignIssues/WebServices.html].

[Bet2008] BETHER, *C. Software service composition in next generation networking environments; Dissertation;* TU-Berlin Berlin, 2008 [http://opus.kobv.de/tuberlin/volltexte/2008/2078/pdf/bether_carsten.pdf].

[BilSin2004] BILGIN, A.S.; SINGH, M.P. A DAML-Based Repository for QoS-Aware Semantic Web Service Selection. In *IEEE International Conference on Web Services (ICWS'04)*: *San Diego, California, 6-9 July 2004,* p 368. IEEE Computer Society; IEEE Computer Society Press: Los Alamitos, Calif., 2004 [http://portal.acm.org/citation.cfm?id=1009386.1010177&coll=Portal&dl=GUIDE&CFID=59898292&CFTOKEN=52577230].

[Boh2009] BOHN, *H. Web Service Composition for Embedded Systems: WS-BPEL extension for DPWS; Univ., Diss.--Rostock, 2008.*, 1st ed.; Sierke: Göttingen, 2009 (ISBN 9783868441086).

[BooHaa et al.2004] BOOTH, DAVID; HAAS, HUGO; MCCABE, FRANCIS; NEWCOMER, ERIC; CHAMPION, MICHAEL; FERRIS, CHRIS; ORCHARD, *David.Web Services Architecture: W3C Working Group Note 11 February 2004*, W3C 08.02.2004 [http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/] (accessed 21. Mar. 2009).

[BroCan et al.2004] BROGI, A.; CANAL, C.; PIMENTEL, E.; VALLECILLO, A. Formalizing Web Service Choreographies. Proceedings of the First International Workshop on Web Services and Formal Methods (WSFM 2004). **2004**, *105* pp. 73–94 [doi:10.1016/j.entcs.2004.05.007].

[BroGos2010] BROCK, M.; GOSCINSKI, A. Toward Ease of Discovery, Selection and Use of Clusters within a Cloud. *Cloud Computing, IEEE International Conference on* **2010** pp. 289–296 [http://doi.ieeecomputersociety.org/10.1109/CLOUD.2010.39].

[BroUro et al.2009] BROMURI, S.; UROVI, V.; MORGE, M.; STATHIS, K.; TONI, *F. A multi-agent system for service discovery, selection and negotiation.* In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*: *AAMAS 2009.* May 10-15, 2009. Decker, K. S., Sichman, J. S., Simão, C., Castelfranchi, C., [Eds.]. 2; International Foundation for Autonomous Agents and Multiagent Systems: Budapest, Hungary, 2009, pp 1395–1396 [www.argugrid.eu/documentation/AAMASDemoBromuri.pdf].

[BruBus et al.2005] BRUIJN, JOS DE; BUSSLER, CHRISTOPH; DOMINGUE, JOHN; FENSEL, DIETER; HEPP, MARTIN; KELLER, UWE; KIFER, MICHAEL; KÖNIG-RIES, BRIGITTA; KOPECKÝ, JACEK; LARA, RUBÉN; LAUSEN, HOLGER; OREN, EYAL; POLLERES, AXEL; ROMAN, DUMITRU; SCICLUNA, JAMES; STOLLBERG, *Michael.Web Service Modeling Ontology (WSMO): W3C Member Submission*, W3C 13.06.2005 [http://www.w3.org/Submission/WSMO/] (accessed 14. Apr. 2010).

[BrzRek et al.2010] BRZOSTOWSKI, K.; REKUĆ, W.; SOBECKI, J.; SZCZUROWSKI, L. Service Discovery in the SOA System. In *Intelligent Information and Database Systems*. Nguyen, N., Le, M., Swiatek, J., [Eds.], *5991*, pp 29–38. Lecture Notes in Computer Science, LNCS Springer Berlin / Heidelberg: 2010 [http://dx.doi.org/10.1007/978-3-642-12101-2_4].

[CabDom et al.2006] CABRAL, LILIANA; DOMINGUE, JOHN; GALIZIA, STEFANIA; GUGLIOTTA, ALESSIO; TANASESCU, VLAD; PEDRINACI, CARLOS; NORTON, *Barry.IRS-III: A Broker for Semantic Web Services based Applications, Knowledge Media Institute, The Open University, Milton Keynes, UK 24.08.2006] (accessed 10. Jun. 2010).*

[Cai2009] CAIRE, *Giovanni.ADEProgramming-Tutorial-for-beginners. 2009 [*http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf] (accessed 13. Aug. 2010).

[Cai2010] CAIRE, *Giovanni.WADE-User-Guide.doc*, Telecom Italia, former CSELT 07.07.2010 [http://jade.tilab.com/wade/doc/WADE-User-Guide.pdf] (accessed 13. Aug. 2010).

[Car2009] CARLSSON, *M. SICStus Prolog User's Manual*, Release 4.0.7, Swedish Institute of Computer Science Kista, Sweden, 18.04.2009.

[ÇelElç2008] ÇELIK, D.; ELÇI, A. Semantic QoS Model for Extended IOPE Matching and Composition of Web Services. In *32nd annual IEEE international computer software and applications, 2008*: *COMPSAC '08*, pp 993–998. IEEE Computer Society; IEEE: Piscataway, NJ, 2008 [http://dx.doi.org/10.1109/COMPSAC.2008.208].

[CEN2011] CEN, CENELEC AND ETSI.*Framework for Smart Grid Architecture Models (SGAM): M/490 Reference Architecture WG*, CEN, CENELEC and ETSI 2011 [http://www.pointview.com/data/files/1/473/2185.pdf] (accessed 25. Feb. 2013).

[ChMeGh2010] CHAINBI, W.; MEZNI, H.; GHEDIRA, *K. PECoDiM: An Agent Based Framework for Autonomic Web Services.* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 543–550 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.94].

[ChrFer et al.2004] CHRISTENSEN, ERIK; FERGUSON, DONALD; FREY, JEFFREY; HADLEY, MARC; KALER, CHRIS; LANGWORTHY, DAVID E.; LEYMANN, FRANK; LOVERING, BRAD; LUCCO, STEVE; MILLET, STEVE; MUKHI, NIRMAL; NOTTINGHAM, MARK; ORCHARD, DAVID; SHEWCHUK, JOHN; SINDAMBIWE, EUGÈNE; STOREY, TONY; WEERAWARANA, SANJIVA; WINKLER, *Steve.Web Services Addressing (WS-Addressing). 05.08.2004 [*http://www.w3.org/Submission/ws-addressing/] (accessed 18. Sep. 2010).

[CuEhRo2002] CURBERA, F.; EHNEBUSKE, D.; ROGERS, *D.Using WSDL in a UDDI Registry: UDDI Best Practise (Version 1.07). May 2002 [*http://www.uddi.org/pubs/wsdlbestpractices.pdf].

[DAI2008] DAI LABOR, TU BERLIN.*JIAC V*, DAI Labor, TU Berlin 2008 [http://jiac.de/?id=35] (accessed 10. Feb. 2011).

[Dal2002a] DALE, *Jonathan.FIPA ACL Message Structure Specification: FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS; Standard SC00061G*, FIPA 03.12.2002 [http://www.fipa.org/specs/fipa00061/SC00061G.pdf] (accessed 13. Aug. 2010).

[Dal2002b] DALE, *Jonathan.FIPA Quality of Service Ontology Specification*, Foundation for Intelligent Physical Agents, FIPA 2002 [http://www.fipa.org/specs/fipa00094/SC00094A.pdf] (accessed 29. Aug. 2010).

[Dal2005] DALE, *Jonathan.FIPA Specifications*, FIPA 2005 [www.fipa.org/specifications/index.html] (accessed 10. Jun. 2012).

[Dal2011] DALE, *Jonathan.Welcome to the Foundation for Intelligent Physical Agents. 06.01.2011 [*http://www.fipa.org/] (accessed 13. Jan. 2011).

[Dee2011] DEEPA.*EventStudio System Designer 4.0: Sequence Diagram Based Modeling; Multiple Scenario Modeling*, EventHelix.com Inc. 2011 [http://eventhelix.com/EventStudio/EventStudio_System_Designer_Brochure.PDF] (accessed 20. Sep. 2011).

[DeuKli2002] DEURSEN, A. VAN; KLINT, P. Domain-Specific Language Desing Requires Feature Descriptions. *Journal of Computing and Information Technology - CIT* **2002** (10) pp. 1–17 [http://hrcak.srce.hr/cit_ojs/index.php/CIT/article/viewFile/1465/1169].

[DicWoo2005] DICKINSON, IAN; WOOLDRIDGE, *Michael.Agents are not (just) web services : considering BDI agents and web services. 2005 [*http://www.hpl.hp.com/techreports/2005/HPL-2005-123.pdf] (accessed 27. Jan. 2010).

[DomCab et al.2004] DOMINGUE, J.; CABRAL, L.; HAKIMPOUR, F.; SELL, D.; MOTTA, *E. IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services.* In *WIW 2004*: *WSMO Implementation Workshop 2004.* Frankfurt, September 29-30, 2004. Bussler, C., Fensel, D., Lausen, H., Oren, E., [Eds.]. Workshop Proceedings 113; CEUR: 2004 [http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-113/paper3.pdf].

[DonHal et al.2004] DONG, X.; HALEVY, A.; MADHAVAN, J.; NEMES, E.; ZHANG, *J. Similarity Search for Web Services.* In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*: *Toronto, Canada, August 31 - September 3, 2004.* Nascimento, M. A., [Ed.]. VLDB Endowment; Morgan Kaufmann: St. Louis, Mo., 2004, pp 372–383 [http://portal.acm.org/citation.cfm?id=1316723#].

[ElfLay2002] ELFATATRY, P.; LAYZELL, A. Software as a service negotiation perspective. Computer Software and Application Conference, 26-29 August. *COMPSAC Proceedings* **2002**, *26* pp. 501–506.

[EroMou2013] EROL-KANTARCI, M.; MOUFTAH, H.T. Smart grid forensic science: applications, challenges, and open issues. *Communications Magazine, IEEE* **2013**, *51* (1) pp. 68–74 [http://ieeexplore.ieee.org/ielx5/35/6400427/06400441.pdf?tp=&arnumber=6400441&isnumber=6400427].

[Eur2009] EUROPEAN FUTURE INTERNET PORTAL.*Home: Future Internet*, European Future Internet Portal 2009 [http://future-internet.eu/] (accessed 18. Mar. 2009).

[Eur2013] EUROPEAN COMMISSION.*Single market for gas & electricity*, European Commission 2013 [http://ec.europa.eu/energy/gas_electricity/index_en.htm] (accessed 25. Feb. 2013).

[EvaFil2007] EVANS, J.; FILSFILS, *C. Deploying IP and MPLS QoS for multiservice networks: Theory and practice.* The Morgan Kaufmann series in networking; Morgan Kaufmann/Elsevier: Amsterdam, 2007 (ISBN 0123705495).

[FarLau2007a] FARRELL, JOEL; LAUSEN, *Holger.Semantic Annotations for WSDL and XML Schema: W3C Recommendation 28 August 2007. 27.08.2007 [*http://www.w3.org/TR/sawsdl/] (accessed 24. Aug. 2010).

[FarLau2007b] FARRELL, JOEL; LAUSEN, *Holger.Semantic Annotations for WSDL and XML Schema (SAWSDL): W3C Recommendation 28 August 2007*, W3C 27.08.2007 [http://www.w3.org/TR/sawsdl/] (accessed 30. Aug. 2010).

[FeiJey2009]	FEINGOLD, MAX; JEYARAMAN, *Ram.WS-Coordination v1.2: OASIS Standard; 2 February 2009*, OASIS 2009 [http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.html] (accessed 06. Dec. 2010).
[FeKeZa2008]	FENSEL, D.; KERRIGAN, M.; ZAREMBA, *M. Implementing Semantic Web Services: The SESA Framework*, 1st ed.; Springer; Springer-Verlag Berlin Heidelberg: Berlin, Heidelberg, 2008 (ISBN 3540770194).
[FenBus2002]	FENSEL, DIETER; BUSSLER, *Christoph.The Web Service Modeling Framework WSMF*, Vrije Universiteit Amsterdam (VU); Oracle Corporation 2002 [http://www.wsmo.org/papers/publications/wsmf.paper.pdf] (accessed 04. Nov. 2009).
[Fra2007]	FRAUNHOFER IIS.*JINI*, Fraunhofer IIS 2007 [http://www.iis.fraunhofer.de/EN/bf/ec/dm/jini.jsp] (accessed 27. Aug. 2010).
[FraDie et al.2005]	FRANZ BAADER, DIEGO CALVANESE, DEBORAH L. MCGUINNESS, DANIELE NARDI, PETER F. PATEL-SCHNEIDER, BAADER, *F., Eds.: The description logic handbook: Theory, implementation, and applications;* Cambridge University Press; Cambridge Univ. Press: Cambridge, 2005,
[FuHao et al.2010]	FU, J.; HAO, W.; TU, M.; MA, B.; BALDWIN, J.; BASTANI, *F.B. Virtual Services in Cloud Computing.* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 467–472 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.125].
[FujSud2009]	FUJII, K.; SUDA, T. Semantics-based context-aware dynamic service composition. *ACM Trans. Auton. Adapt. Syst.* **2009**, *4* (2) pp. 1–31 [http://doi.acm.org/10.1145/1516533.1516536].
[GaRuRu2010]	GARCÍA, J.; RUIZ, D.; RUIZ-CORTÉS, *A. A Model of User Preferences for Semantic Services Discovery and Ranking.* In *The Semantic Web: Research and Applications*: *7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010 ; proceedings, part II.* Aroyo, L., Antoniou, G., Hyvönen, E., Teije, A. ten, Stuckenschmidt, H., Cabral, L., Tudorache, T., [Eds.]. Lecture notes in computer science 6089; Springer Berlin / Heidelberg; Springer: Berlin, 2010, *6089*, pp 1–14 [http://dx.doi.org/10.1007/978-3-642-13489-0_1].
[Ger2006]	GERSTBACH, *Peter.ebXML vs. Web Services: Comparison of ebXML and the Combination of SOAP/WSDL/UDDI/BPEL*, Vienna University of Technology 23.02.2006 [http://www.gerstbach.at/2006/ebxml-ws/ebxml-ws.pdf] (accessed 03. Dec. 2010).

[GraMax et al.2010] GRANDISON, T.; MAXIMILIEN, M.E.; THORPE, S.; ALBA, *A. Towards a Formal Definition of a Computing Cloud.* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 191–192 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.111].

[Gri2007] GRIMM, S. Discovery. Identifying Relevant Services In *Semantic Web Services*: *Concepts, Technologies, and Applications.* Studer, R., Grimm, S., Abecker, A., [Eds.], pp 211–244. Springer-Verlag GmbH; Springer: Berlin, Heidelberg, 2007.

[Gri2010] GRIMSHAW, *David.JADE Administration Tutorial. 21.04.2010 [*http://jade.tilab.com/doc/tutorials/JADEAdmin/jadeArchitecture.html] (accessed 13. Aug. 2010).

[Gru2009] GRUBER, T. Ontology (Computer Science). definition In *Encyclopedia of database systems.* Liu, L., Özsu, M. T., [Eds.]. Springer reference Springer: New York, NY, 2009 [http://tomgruber.org/writing/ontology-definition-2007.htm].

[GrüMen2003] GRÜNINGER, M.; MENZEL, C. The Process Specification Language (PSL). Theory and Applications. *AI magazine* **2003**, *24* (4) pp. 63–74.

[GudHad et al.2007] GUDGIN, MARTIN; HADLEY, MARC; MENDELSOHN, NOAH; MOREAU, JEAN-JACQUES; NIELSEN, HENRIK FRYSTYK; KARMARKAR, ANISH; LAFON, *Yves.SOAP Version 1.2 Part 1: Messaging Framework (Second Edition),* W3C 27.04.2007 [http://www.w3.org/TR/soap12-part1/] (accessed 29. Mar. 2009).

[GunSah et al.2012] GUNGOR, V.C.; SAHIN, D.; KOCAK, T.; ERGUT, S.; BUCCELLA, C.; CECATI, C.; HANCKE, G.P. Smart Grid and Smart Homes: Key Players and Pilot Projects. *Industrial Electronics Magazine, IEEE* **2012**, *6* (4) pp. 18–34 [http://ieeexplore.ieee.org/ielx5/4154573/6378501/06378528.pdf?tp=&arnumber=6378528&isnumber=6378501].

[GurZei2005] GURGUIS, S.A.; ZEID, A. Towards autonomic web services: achieving self-healing using web services. *SIGSOFT Softw. Eng. Notes* **2005**, *30* pp. 1–5 [doi:10.1145/1082983.1083069].

[GuZaRo2009] GUAN, T.; ZALUSKA, E.; ROURE, D. de. An autonomic service discovery mechanism to support pervasive devices accessing the semantic grid. *International Journal of Autonomic Computing* **2009**, *1* (1) pp. 34–49 [http://inderscience.metapress.com/link.asp?id=70744701427n3072].

[Hab2009] HABALA, O. Semantically-Aided Data-Aware Service Workflow Composition. *Lecture Notes in Computer Science (LNCS)* **2009** (No. 5404 (2009)) pp. 317–328.

[HaMaKü2010] HASEEB, A.; MATSKIN, M.; KÜNGAS, P. Distributed Web Services Discovery Middleware for Edges of Internet. *Web Services, IEEE International Conference on* **2010**, *0* pp. 680–682 [http://doi.ieeecomputersociety.org/10.1109/ICWS.2010.87].

[HaReMa2004] HAUSMANN, J.H.; REIKO HECKEL; MARC LOHMANN. Model-based Discovery of Web Services. *Web Services, IEEE International Conference on* **2004** pp. 324-324 [http://doi.ieeecomputersociety.org/10.1109/ICWS.2004.1314754].

[HarSur2004] HARTMANN, J.; SURE, Y. An Infrastructure for Scalable, Reliable Semantic Portals. *IEEE Intelligent Systems* **2004**, *19* (3) pp. 58–65 [http://portal.acm.org/citation.cfm?id=998482.998552&coll=Portal&dl=GUIDE&CFID=59898292&CFTOKEN=52577230].

[Hou1996] HOUSTON, *P. J.Intoduction to DCE an Encina: Whitepaper*, Transarc Corp. 1996 [http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/Whitepapers/].

[Hua2008] HUANG, *X. Semantic Agent Support for Managed Open Information Retrieval Services; Ph.D.-Thesis;* Queen Mary, University of London London, UK, 25. Jan. 2008 [http://www.elec.qmul.ac.uk/networks/documents/XuanHuang-PhDThesisFinal_000.pdf].

[HuaZha et al.2010] HUANG, J.; ZHANG, Y.; YEN, I.-L.; CARSON, J.T.; SIOK, M.F.; BASTANI, F.; ZHAO, Y.; DONG, *J. Real-Time Service-Oriented Distributed Governance.* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 479–484 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.100].

[Huh2006] HUHNS, *M. A Research Agenda for Agent-Based Service-Oriented Architectures.* In *Cooperative Information Agents X*: *10th International Workshop, CIA 2006, Edinburgh, UK, September 11-13, 2006, Proceedings.* Klusch, M., Payne, T. R., Rovatsos, M., [Eds.]. Lecture Notes in Artificial Intelligence, LNAI 4149; Springer-Verlag GmbH: Berlin Heidelberg, 2006, pp 8–22.

[HuhSte2000] HUHNS, M.N.; STEPHENS, L.M. Multiagent Systems and Societies of Agents. In *Multiagent Systems*: *A Modern Approach to Distributed Artificial Intelligence.* Weiss, G., [Ed.], pp 79–120. MIT Press: Cambridge, Massachusetts, 2000.

[IBM2006] IBM.*developerWorks Interviews: Tim Berners-Lee*, IBM 2006 [http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html] (accessed 29. Mar. 2009).

[IBM2007a] IBM.*WS-BPEL Extension for People*, IBM 2007 [http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/] (accessed 07. Dec. 2010).

[IBM2007b] IBM CORPORATION.*Business Process Execution Language for Web Services version 1.1: BPEL4WS (BPEL)*, IBM Corporation 08 Feb 2007 [http://www.ibm.com/developerworks/library/specification/ws-bpel/] (accessed 02. Apr. 2009).

[IBM2009] IBM.*New to Web services*, IBM 2009 [http://www.ibm.com/developerworks/webservices/newto/websvc.html?S_TACT=105AGX28&S_CMP=DLMAIN] (accessed 24. Mar. 2009).

[IEC2007] IEC.*H.323 (White Paper): On-Line Education: Tutorial*, IEC 2007 [http://www.iec.org/online/tutorials/h323/index.asp] (accessed 29. Mar. 2009).

[Int1995] *ODP Trading Function*, DIS 13235; Draft International Standard, International Organisation for Standardization (ISO), International Electrotechnical Commission (IEC), Juni 1995 [ftp://ftp.fhg.de/archive/gmd/div/documents/iso/RM-ODP/…/trader.ps.gz].

[ISO2004] ISO.*Process Specification Language (PSL): International Standard 18629 of the International Standards Organization (ISO),; ISO TC 184 (Industrial automation systems and integration)*, ISO 2004 [http://www.mel.nist.gov/psl/] (accessed 06. Dec. 2010).

[ISOITU1996] ISO; ITU.*Open Distributed Processing - Reference Model: ISO/IEC IS 10746 | ITU-T X.900*, ISO; ITU 1996 [http://www.joaquin.net/ODP/index.html] (accessed 30. Mar. 2009).

[ITU1997] ITU.*Information technology – Open Distributed Processing: Reference Model: Overview; X901-X SERIES: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY*, ITU 1997 [http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.901] (accessed 10. Aug. 2011).

[ITUISO1997a] ITU; ISO. *ODP Trading Function: Part 1 - 3 (Specification, Test Cases, ... using OSI Directory).* ISO/IEC IS 13235-x, ITU-T Draft Rec. X950-x, 1997,

[ITUISO1997b] ITU; ISO. *Open Distributed Processing (ODP): (Interface Definition Language and Type Repository Function).* ISO/IEC (IS 14570 and CD14769) ITU/T Draft Rec. X9nn, 1997,

[JacMud1996] JACOB, B.L.; MUDGE, T. The trading function in action. In *Proceedings of the 7th ACM SIGOPS European Workshop*: *Systems Support for Worldwide Applications.* Herbert, A., Tanenbaum, A. S., [Eds.], pp 241–247. ACM Press: Connemara, Ireland, 1996.

[JacZav1998]	JACKSON, M.; ZAVE, P. Distributed Feature Composition - A Virtual Architecture for Telecommunications Services. IEEE Transactions on Software Engineering. *Software Engineering, IEEE Transactions on* **1998**, *24* (10) pp. 831–847 [http://ieeexplore.ieee.org/ielx4/32/15729/00729683.pdf?tp=&arnumber=729683&isnumber=15729].
[Jin2010]	JINI.ORG.*Jini: Main Page*, Jini.org 24.11.2010 [http://www.jini.org/wiki/Main_Page] (accessed 03. Dec. 2010).
[Joy2005]	JOYNER, *Ian.Open Distributed Processing (ODP): Unplugged! 06.12.2005 [*http://homepages.tig.com.au/~ijoyner/ODPUnplugged.html] (accessed 04. Apr. 2009).
[Kaa2003a]	KAARTHIK.*METEOR-S Web Service Discovery Infrastructure (MWSDI)*, University of Georgia 2003 [http://lsdis.cs.uga.edu/proj/meteor/mwsdi.html] (accessed 08. Dec. 2010).
[Kaa2003b]	KAARTHIK.*METOER-S Web Service Composition Framework (MWSCF)*, University of Georgia 2003 [http://lsdis.cs.uga.edu/proj/meteor/mwscf/mwscf.html] (accessed 08. Dec. 2010).
[KaBuRi2004]	KAVANTZAS, NICKOLAOS; BURDETT, DAVID; RITZINGER, *Gregory.Web Services Choreography Description Language Version 1.0: W3C Working Draft 27 April 2004*, W3C 27.04.2004 [http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/] (accessed 22. Jul. 2009).
[KavBur et al.2005]	KAVANTZAS, NICKOLAOS; BURDETT, DAVID; RITZINGER, GREGORY; FLETCHER, TONY; LAVON, YVES; BARRETO, *Charlton.Web Services Choreography Description Language Version 1.0: WS-CDL; W3C Candidate Recommendation 9 November 2005*, W3C 09.11.2005 [http://www.w3.org/TR/ws-cdl-10/] (accessed 03. Dec. 2010).
[KaWaHu2007]	KANG, Z.; WANG, H.; HUNG, P.C. WS-CDL+ for web service collaboration. *Information Systems Frontiers* **2007**, *9* pp. 375–389 [http://portal.acm.org/citation.cfm?id=1285880.1285884&coll=Portal&dl=GUIDE&CFID=59898292&CFTOKEN=52577230].
[KiBeSt2007]	KIEFER, C.; BERNSTEIN, A.; STOCKER, *M. The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks.* In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007.* ISWC 2007 + ASWC 2007, Busan, Korea, November 11 - 15, 2007 ; proceedings. Aberer, K., [Ed.]. 4825; Springer; SpringerLink [host]: Berlin, 2007, pp 295–309.

[KiLaWu1995] KIFER, MICHAEL; LAUSEN, GEORG; WU, *James.Logical Foundations of Object Oriented and Frame Based Languages. 1995 [*http://www.cs.umbc.edu/courses/771/papers/flogic.pdf] (accessed 08. Dec. 2010).

[KlFrSy2006] KLUSCH, M.; FRIES, B.; SYCARA, *K. Automated semantic web service discovery with OWLS-MX.* In *International Conference on Autonomous Agents*: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems.* Hakodate, Japan. ACM Press: New York, NY, USA, 2006, *SESSION: Ontologies and web services,* pp 915–922 [http://portal.acm.org/citation.cfm?id=1160633.1160796&coll=Portal&dl=GUIDE&CFID=59898292&CFTOKEN=52577230].

[KlHoSc2000] KLOOS, R.; HOFFMANN, M.; SCHROEDER, M. Intelligent Traders and Aspects of Security in Co-Operative Rooms. In *Deutsche-CSCW Konferenz 2000*: *workshop: Agents and CSCW: A Fruitful Marriage?* Branki, C., Newman, J., Unland, R., [Eds.]. München, 2000.

[KlKaZi2009] KLUSCH, M.; KAPAHNKE, P.; ZINNIKUS, I. Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer. In *The semantic web*: *Research and applications ; 6. European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31 - June 4, 2009 ; proceedings.* Aroyo, L., [Ed.], pp 550–564. 5554; Springer-Verlag; Springer: Heraklion, Crete, Greece, 2009.

[KlReSc2001] KLOOS, R.; REINEMA, R.; SCHROEDER, M. UNITE - Modern Teamwork with Adaptive Communications. In *Software Technology Outreach*: *E-Working: How to Improve Effectiveness.* Canning, A., Stock, T., [Eds.]. London, 2001.

[KlReSc2002] KLOOS, R.; REINEMA, R.; SCHROEDER, M. Adaptive Traders for Communication in Cooperative Rooms. *International Journal of Information Technology & Decision Making (IJITDM)* **September 2002**, *Volume 1* (3) pp. 401–421 [http://www.worldscinet.com/cgi-bin/details.cgi?id=jsname:ijitdm&type=all].

[KlScRe2000] KLOOS, R.; SCHROEDER, M.; REINEMA, R. Intelligent Traders for Communication in Coorperative Rooms. In *Autonomous Agents 2000*: *Workshop 8: Intelligent Agents for CSCW: Technology and Risks.* Petsch, M., Lees, B., [Eds.]. ACM Press: Barcelona, Spain, 2000.

[Klu2000a] KLUSCH, M. Intelligent Information Agents. In *Proceedings of the EASSS2000*: *2. European Agent System Summer School.* Klusch, M., [Ed.]. Saarbrücken, 2000.

[Klu2000b] KLUSCH, *M., Ed.: Proceedings of the EASSS2000: 2. European Agent System Summer School.* Saarbrücken, 2000,

[Klu2008] KLUSCH, *Matthias.OWL-S and SAWSDL Service Matchmakers: s3c-2008. 19.11.2008 [*http://www-ags.dfki.uni-sb.de/~klusch/s3/s3c-2008.pdf] (accessed 18. Sep. 2011).

[KluFri et al.2009] KLUSCH, M.; FRIES, B.; KATIA SYCARA; SYCARA, K. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. *Web Semant* **2009**, *7* (2) pp. 121–133 [http://www-ags.dfki.uni-sb.de/~klusch/papers/owlsmx-08-final.pdf].

[KluKau2009] KLUSCH, M.; KAUFER, F. WSMO-MX: A hybrid Semantic Web service matchmaker. *Web Intelligence and Agent Systems* **2009**, *7* (1) pp. 23–42 [http://dx.doi.org/10.3233/WIA-2009-0153].

[KlUnBr2009] KLOOS, R.; UNLAND, R.; BRANKI, *C. - ACTAS - Adaptive Composition and Trading Based on Agents.* In *Proceedings of the Fifth International Workshop on Modeling of Objects, Components and Agents*: *MOCA '09, Hamburg.* Duvigneau, M., Moldt, D., [Eds.]. Bericht FBIUniHHab2006: Hamburg, 2009, pp 151–171.

[KlUnBr2010] KLOOS, R.; UNLAND, R.; BRANKI, *C. Service Discovery with Semantic Characteristics: ACTAS.* In *2010 6th World Congress on Services*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* IEEE Computer Society: Los Alamitos, Calif, 2010, pp 551–558 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.106].

[KlUnBr2012] KLOOS, R.; UNLAND, R.; BRANKI, *C. Liberation of smart grids addressed through SOC: An application of the Service Discovery framework ACTAS.* In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on.* IEEE; Piscataway, July 2012, pp 1056–1061.

[Kno2003] KNORR, *Eric.2004: The Year of Web Services: CIO Insider*, CIO 2003 [http://www.cio.com/article/32050/2004_The_Year_of_Web_Services] (accessed 25. Jan. 2011).

[Ko2009] KO, R.K.L. A computer scientist's introductory guide to business process management (BPM). *Crossroads* **2009**, *15* (4) pp. 11–18 [doi:10.1145/1558897.1558901].

[Kos1999] KOSTKOVA*, P. MAGNET: A Dynamic Resource Managment Architecture; The City University, London, UK, Department of Computing, thesis of dissertation;* City University , London, July 1999.

[KosMar et al.2008] KOSAR, T.; MARTINEZ-LOPEZ, P.E.; BARRIENTOS, P.A.; MERNIK, M.; KOSAR, T.; MARTINEZLOPEZ, P.; BARRIENTOS, P.; MERNIK, M. A preliminary study on various implementation approaches of domain-specific language. *Information and Software Technology* **2008**, *50* (5) pp. 390–405.

[Kot1988] KOTLER, *P. Marketing Management Analysis, Planning, Implementation, and Control,* 6th ed.; Prentices-Hall International: 1988,

[Kot2003] KOTLER, *P. Marketing Management*, 11th ed.; Prentices-Hall International: Upper Saddle River, NJ, USA, 2003 (ISBN Ko).

[KrKrKu2009] KREBS, M.; KREMPELS, K.-H.; KUCAY, M. A Unified Service Discovery Architecture for Wireless Mesh Networks. In *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*: *7th International IFIP-TC6 Networking Conference Singapore, May 5-9, 2008 Proceedings.* Das, A., Pung, H., Lee, F., Wong, L., Lee, F. B. S., Pung, H. K., Wong, L. W. C., [Eds.], *4982,* pp 865–876. Lecture notes in computer science 4982; Springer Berlin / Heidelberg; Springer-Verlag Berlin Heidelberg: Berlin, Heidelberg, 2009 [http://dx.doi.org/10.1007/978-3-540-79549-0_76].

[KumNan2005] KUMARAN, SANTHOSH; NANDI, *Prabir.Dynamic e-Business Using BPEL4WS, WS-Coordination, WS-Transaction, and Conversation Support for Web Services: IBM Research to Conversation Support*, IBM Research 2005 [http://www.research.ibm.com/convsupport/papers/BPEL%20&%20Conversations.htm].

[KünMat2006] KÜNGAS, P.; MATSKIN, M. Semantic Web Service Composition Through a P2P-Based Multi-agent Environment. *Agents and Peer-to-Peer Computing (LNCS)* **2006** (4118) pp. 106–119 [http://www.springerlink.com/content/e653072k2466g627/].

[KvaRon et al.2005] KVALØY, T.A.; RONGEN, E.; TIRADO-RAMOS, A.; SLOOT, P. Automatic Composition and Selection of Semantic Web Services. In *Advances in Grid Computing - EGC 2005*: *European Grid Conference, Amsterdam, The Netherlands, February 14-16, 2005, Revised Selected Papers.* Sloot, P. M. A., Hoekstra, A. G., Priol, T., Reinefeld, A., Bubak, M., Bubak, M., Hoekstra, A. G., Priol, T., Reinefeld, A., Sloot, P. M. A., [Eds.], *3470,* pp 184–192. Lecture Notes in Computer Science, LNCS 3470; Springer Berlin / Heidelberg; Springer-Verlag GmbH: Berlin Heidelberg, 2005 [http://dx.doi.org/10.1007/11508380_20].

[LaPaSt2003] LA, V.Q.; PATTERSON, P.G.; STYLES, C.W. Determinants of Export Performance Across Service Types - A Conceptual Model. In *School of Marketing Working Paper, 5.* University of New South Wales: Sydney, Australia, 2003.

[LauLar et al.2007] LAUSEN, H.; LARA, R.; POLLERES, A.; BRUIJN, J.D.; ROMAN, D. Semantic Annotation for Web Services. In *Semantic Web Services*: *Concepts, Technologies, and Applications.* Studer, R., Grimm, S., Abecker, A., [Eds.]. Springer-Verlag GmbH; Springer: Berlin, Heidelberg, 2007.

[LeKiKa2010] LE NGAN, D.; KIRCHBERG, M.; KANAGASABAI, R. Review of Semantic Web Service Discovery Methods. *Services, IEEE Congress on* **2010** pp. 176–177 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.85].

[LiaRod2013] LIANG ZHOU; RODRIGUES, J.J.P.C. Service-oriented middleware for smart grid: Principle, infrastructure, and application. *Communications Magazine, IEEE* **2013**, *51* (1) pp. 84–89 [http://ieeexplore.ieee.org/ielx5/35/6400427/06400443.pdf?tp=&arnumber=6400443&isnumber=6400427].

[LiGaSh2010] LIU, S.; GANG QUAN; SHANGPING REN. On-Line Scheduling of Real-Time Services for Cloud Computing. *Services, IEEE Congress on* **2010** pp. 459–464 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.109].

[Lov1983] LOVELOCK, C.H. Classifying Services to Gain Strategic Marketing Insights. In *Journal of Marketing, 47 (3),* pp 9–20. American Marketing Association: 1983.

[Lov1988] LOVELOCK, C.H. Classifying Services to gain Strategic Marketing Insights. In *Managing Services*: *Marketing, operations and human resources.* Lovelock, C. H., [Ed.]. Prentice-Hall International; Longham Higher Education: London; Englewood Cliffs, 1988.

[LutMic2007] LUTZ; MICHAEL. Ontology-Based Descriptions for Semantic Discovery and Composition of Geoprocessing Services. *GeoInformatica* **2007**, *11* (1) pp. 1–36 [http://dx.doi.org/10.1007/s10707-006-7635-9].

[MahSpa2010] MAHBUB, K.; SPANOUDAKIS, G. Proactive SLA Negotiation for Service Based Systems. *Services, IEEE Congress on* **2010** pp. 519–526 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.15].

[ManMcI2003] MANDELL, D.J.; MCILRAITH, S.A. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In *The Semantic Web - ISWC 2003*: *Second International Semantic Web Conference ; proceedings.* Fensel, D., Sycara, K. P., Mylopoulos, J., [Eds.], pp 227–241. Lecture Notes in Computer Science, LNCS 2870; Springer: Berlin, 2003.

[Mar2003] MARTIN OWEN AND JOG RAJ, *Popkin Software.BPMN and Business Process Management. 29.08.2003 [*http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf] (accessed 29. Aug. 2010).

[Mar2007] MARCHESE, *M. QoS over heterogeneous networks;* Wiley: Chichester, 2007 (ISBN 9780470017524).

[MarBur et al.2008] MARTIN, DAVID; BURSTEIN, MARK; HOBBS, JERRY R.; LASSILA, ORA; MCDERMOTT, DREW V.; MCILRAITH, SHEILA; NARAYANAN, SRINI; PAOLUCCI, MASSIMO; PARSIA, BIJAN; PAYNE, TERRY; SIRIN, EVREN; SRINIVASAN, NAVEEN; SYCARA, *Katia.OWL-S 1.2 Release. 18.11.2008 [*http://www.ai.sri.com/daml/services/owl-s/1.2/] (accessed 18. Nov. 2009).

[MarMer et al.2001] Marvie, R.; Merle, P.; Geib, J.; Leblanc, S. 3.3.8 TORBA: Trading Contracts for CORBA. In *6th USENIX Conference on Object-Oriented Technologies and Systems*. The USENIX Association: San Antonio, Texas, USA, 2001.

[MarPim2010] Martín, J.A.; Pimentel, *E. Feature-Based Discovery of Services with Adaptable Behaviour*. In *8th Ieee european conference on web services, ECOWS'10*: *Ayia Napa, Cyprus, 1-3 December 2010*. Brogi, A., Pautasso, C., Papadopoulos, G. A., [Eds.]. IEEE Computer Society Press: Los Alamitos CA, 2010.

[MedBou2005] Medjahed, B.; Bouguettaya, A. A Dynamic Foundational Architecture for Semantic Web Services. *Distributed and Parallel Databases* **2005**, *17* (2) pp. 179–206 [http://dx.doi.org/10.1007/s10619-004-0190-1].

[MicChi et al.2007] Michael Mrissa; Chirine Ghedira; Djamal Benslimane; Zakaria Maamar; Florian Rosenberg; Schahram Dustdar. A context-based mediation approach to compose semantic Web services. *ACM Trans. Internet Technol.* **2007**, *8* (1) pp. 23, Art. 4 [http://doi.acm.org/10.1145/1294148.1294152].

[ModKem2009] Modi, Vipul; Kemp, *Devon.OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1: OASIS Standard*, OASIS 2009 [http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html] (accessed 20. Feb. 2011).

[MüKoBr2006] Müller, I.; Kowalczyk, R.; Braun, *P. Towards Agent-based Coalition Formation for Service Composition, 2006 [*http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4052901].

[NadGoo et al.2007] Nadalin, Anthony; Goodner, Marc; Gudgin, Martin; Barbir, Abbie; Granqvist, *Hans.WS-Trust 1.3: OASIS Standard; OASIS Web Service Secure Exchange TC*, OASIS 24.03.2007 [http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html] (accessed 19. Feb. 2011).

[NIS2010] NIST (National Institute of Standards and Technology).*Cloud Computing: Nist Defintion of Cloud Computing (v15); Forum and Workshop, 20.05.2010*, NIST (National Institute of Standards and Technology) 27.08.2010 [http://csrc.nist.gov/groups/SNS/cloud-computing/] (accessed 26. Jan. 2011).

[NoSaZa2007] Nomane Ould Ahmed M'Bareck; Samir Tata; Zakaria Maamar. *Towards An Approach for Enhancing Web Services Discovery*. In *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2007*: *WETICE 2007 ; 18 - 20 June 2007, Paris, France ; proceedings*. Reddy, S. M., [Ed.]. IEEE: Piscataway, NJ, 2007, *0*, pp 357–364 [http://doi.ieeecomputersociety.org/10.1109/WETICE.2007.180].

[O'EdHo2005] O'SULLIVAN, JUSTIN; EDMOND, DAVID; HOFSTEDE, *Arthur H. M. ter.Formal description of non-functional service properties*, Queensland University of Technology 2005 [http://www.wsmo.org/papers/OSullivanTR2005.pdf] (accessed 24. Jul. 2011).

[OAS2003] OASIS.*Cover Pages: Business Process Modeling Language (BPML)*, OASIS 29.08.2003 [http://xml.coverpages.org/bpml.html] (accessed 02. Apr. 2009).

[OAS2005] OASIS.*OASIS - Committees - OASIS UDDI Specifications TC: Version 3*, OASIS 2005 [http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3] (accessed 29. Mar. 2009).

[OAS2006a] OASIS.*ebXML - Enabling A Global Electronic Market*, OASIS 2006 [http://www.ebxml.org/] (accessed 03. Dec. 2010).

[OAS2006b] OASIS.*OASIS Web Services Security (WSS) TC: WS-Security 1.1*, OASIS 28.11.2006 [http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss] (accessed 18. Sep. 2010).

[OAS2006c] OASIS.*OASIS Web Services Security (WSS) TC*, OASIS 2006 [http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss] (accessed 07. Dec. 2010).

[OAS2009a] OASIS.*OASIS Web Services Atomic Transaction Specification: (WS-AtomicTransaction); February 2 2009*, OASIS 2009 [http://docs.oasis-open.org/ws-tx/wsat/2006/06] (accessed 06. Dec. 2010).

[OAS2009b] OASIS.*OASIS Web Services Business Activity Specification: (WS-BusinessActivity); February 2 2009*, OASIS 2009 [http://docs.oasis-open.org/ws-tx/wsba/2006/06] (accessed 06. Dec. 2010).

[OAS2009c] OASIS.*OASIS Web Services Transaction (WS-TX) TC*, OASIS 2009 [http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx] (accessed 06. Dec. 2010).

[OAS2010] OASIS.*EBXML | Online Community for Electronic Business Using XML (ebXML) Standards*, OASIS 2010 [http://ebxml.xml.org/] (accessed 03. Dec. 2010).

[OAS2011] OASIS.*Web Services Interoperability (WS-I): Activities and announcements*, OASIS 14.01.2011 [http://www.oasis-ws-i.org/] (accessed 01. Feb. 2011).

[OAUDxm2008] OASIS; UDDI CONSORTIUM; XML-ORG.*OWL-S/UDDI Matchmaker | UDDI*, OASIS; UDDI Consortium; xml-org 12.12.2008 [http://uddi.xml.org/news/owl-suddi-matchmaker] (accessed 13. Dec. 2008).

[ODP 2004] TU-BERLIN/CIS.*Reference Model of Open Distributed Processing - RM-ODP: A short introduction*, TU-Berlin/CIS 07.04.2004 [http://cis.cs.tu-berlin.de/Forschung/Projekte/neweconomy/lernmodule/interoperabilitaet_rmodp/Output/html/a00.html] (accessed 30. Mar. 2009).

[O'EdHo2002] O'SULLIVAN, J.; EDMOND, D.; HOFSTEDE, A.H.M., ter. What's in a service?: Towards accurate description of non-functional service properties. Service Description: A survey ot the general nature of services. *Distributed and Parallel Databases (DAPD)* **2002**, *12* (2/3) pp. 117–133.

[OMG2008] OMG.*CORBA 3.1*, OMG 2008 [http://www.omg.org/spec/CORBA/3.1/] (accessed 29. Mar. 2009).

[OraSUN2005] ORACLE CORPORATION; SUN MICROSYSTEMS.*Jini(TM) Technology Starter Kit Current Releases*, Oracle Corporation; SUN microsystems 2005 [https://starterkit.dev.java.net/downloads/index.html] (accessed 27. Aug. 2010).

[Org2004] *UDDI Executive Overview*, Enabling Service-Oriented Architecture; White Paper, Organization for the Advancement of Structured Information Standards, www.oasis-open.org, October 2004 [http://uddi.xml.org/files/uddi-exec-wp.pdf].

[Org2010] ORGANIC COMPUTING INITIATIVE.*Organic Computing*, Organic Computing Initiative 2010 [http://www.organic-computing.de/] (accessed 26. Jan. 2011).

[OSG2012] OSG.*GS OSG 001 - V1.1.1 - Open Smart Grid Protocol (OSGP)*, ETSI Industry Specification Group (ISG) 2012 [http://www.etsi.org/deliver/etsi_gs/OSG/001_099/001/01.01.01_60/gs_osg001v010101p.pdf] (accessed 25. Feb. 2013).

[PaDaDi2010] PAGANELLI, F.; DAVID PARLANTI; DINO GIULI. Message-Based Service Brokering and Dynamic Composition in the SAI Middleware. *Services Computing, IEEE International Conference on* **2010**, *0* pp. 474–481 [http://doi.ieeecomputersociety.org/10.1109/SCC.2010.87].

[PanTsa2009] PANTAZOGLOU, MICHAEL; TSALGATIDOU, *Aphrodite.The Unified Service Query Language: USQL; Technical Report*, National and Kapodistrian University of Athens, Greece 19.07.2009 [http://www.s3lab.com/usql-tr.pdf] (accessed 31. Aug. 2010).

[Pao2003] PAOLUCCI, M. The DAML-S Virtual Machine. *Lecture notes in computer science* **2003** (2870) pp. 290–305.

[PaoKaw et al.2002] PAOLUCCI, M.; KAWAMURA, T.; PAYNE, T.; SYCARA, *K. Semantic Matching of Web Services Capabilities.* In *The Semantic Web -- ISWC 2002*: *First International Semantic Web Conference Sardinia, Italy, June 9-12, 2002 Proceedings.* Proceedings. Horrocks, I., Hendler, J., [Eds.]. 2342; Springer; Springer-Verlag Berlin Heidelberg: Berlin, Heidelberg, 2002, *2342,* pp 333–347 [http://dx.doi.org/10.1007/3-540-48005-6_26].

[Pee2005] PEER, *Joachim.Semantic Service Markup with SESMA, MCM Institute University of St. Gallen 2005] (accessed 30. Aug. 2010).*

[PeNiHu2009] PENG, H.; NIU, W.; HUANG, R. Similarity Based Semantic Web Service Match. In *Web Information Systems and Mining*: *International conference, WISM 2009, Shanghai, China, November 7 - 8, 2009 ; proceedings.* Liu, W., Luo, X., Wang, F., Lei, J., Wang, F. L., [Eds.], *5854,* pp 252–260. Lecture notes in computer science 5854; Springer Berlin / Heidelberg; Springer: Berlin, 2009 [http://dx.doi.org/10.1007/978-3-642-05250-7_27].

[PisBer et al.2004] PISTORE, M.; BERTOLI, P.; CUSENZA, E.; MARCONI, A.; TRAVERSO, *P.WS-GEN: A Tool for the Automated Composition of Semantic Web Services: IWCS 2004,* University of Trento; ITC-IRST 2004 [http://iswc2004.semanticweb.org/demos/26/paper.pdf] (accessed 05. Apr. 2009).

[PoToTu2007] POGGI, AGOSTINO; TOMAIUOLO, MICHELE; TURCI, *Paola.An Agent-Based Service Oriented Architecture. 04.09.2007 [*http://woa07.disi.unige.it/papers/PoggiSOA.pdf] (accessed 27. Jan. 2010).

[Pre2007] PREIST, C. Goals and Vision. Combining Web Services with Semantic Web Technology In *Semantic Web Services*: *Concepts, Technologies, and Applications.* Studer, R., Grimm, S., Abecker, A., [Eds.], pp 159–178. Springer-Verlag GmbH; Springer: Berlin, Heidelberg, 2007.

[PreByd et al.2001] PREIST, C.; BYDE, A.; BARTOLINI, C.; PICCINELLI, G. Towards Agent-Based Service Composition through Negotiation in Multiple Auctions. In *Proceedings of the AISB'01*: *Symposium on Information Agents for Electronic Commerce.* Schroeder, M., Stathis, K., [Eds.], pp 7–16. AISB´01, Agents & Cognition University of York: Heslington, York, England, 2001.

[PudMar et al.1995] PUDER, A.; MARKWITZ, S.; GUDERMANN, F.; GEIHS, K. AI-based Trading in Open Distributed Environments. In *International Conference on Open Distributed Processing (ICODP'95).* Chapman and Hall: 1995.

[RaKuMa2004]	RAO, J.; KUNGAS, P.; MATSKIN, M. Logic-based Web Services Composition: from Service Description to Process Model. In *IEEE International Conference on Web Services (ICWS'04): San Diego, California, 6-9 July 2004*, pp 446–453. IEEE Computer Society; IEEE Computer Society Press: Los Alamitos, Calif., 2004 [http://dx.doi.org/10.1109/ICWS.2004.71].
[RamHol et al.2009]	RAMBOLD, M.; HOLGER KASINGER; FLORIAN LAUTENBACHER; BERNHARD BAUER. *Towards Autonomic Service Discovery.* In *2009 IEEE International Conference on Services Computing (SCC 2009): Bangalore, India, 21 - 25 September 2009 ; [together with the 2009 IEEE International Conference on Web Services (ICWS 2009), SCC 2009 forms the IEEE Congress on Services (Services 2009)].* IEEE: Piscataway, NJ, 2009, pp 192–201 [http://doi.ieeecomputersociety.org/10.1109/SCC.2009.59].
[ReiBah et al.2000]	REINEMA, R.; BAHR, K.; BURKHARDT, H.-J.; HOVESTADT, L. Cooperative Rooms -- Symbiosis of Real and Virtual Worlds. In *Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis.* Nashville, Texas, 2000.
[SaeJaf2005]	SAEED, M.; JAFFAR-UR-REHMANN, M. Enhancement of software engineering by shifting from software product to software service. Information and Communication Technology, First International Conference 27.-28. August. *ICICT Proceedings* **2005** pp. 302–308.
[SaNaMa2005]	SATTANATHAN, S.; NARENDRA, N. C.; MAAMAR, *Z.ConWeSc - Context-based Semantic Web Services Composition; ICSOC, 05*, National Institute of Technology Karnataka Surathkal, India; IBM Software Labs India Bangalore, India; Zayed University, U.A.E, zakaria 2005 [http://www-rocq.inria.fr/who/Sattanathan.Subramanian/Sattanathan_ICSoC2005.pdf] (accessed 05. Apr. 2009).
[SaNaMa2006]	SATTANATHAN, S.; NARENDRA, N.C.; MAAMAR, Z. Ontologies for Specifying and Reconciling Contexts of Web Services. Proceedings of the First International Workshop on Context for Web Services (CWS 2005). *Electronic Notes in Theoretical Computer Science* **2006**, *146* (1) pp. 43–57 [http://www.sciencedirect.com/science/article/B75H1-4J1J59V-5/2/9fea708d3de87d9c32631e9b5aea9d95].
[Ser2009]	SERVICE WEB 3.0.*Service Web 3.0*, Service Web 3.0 2009 [http://www.serviceweb30.eu] (accessed 18. Mar. 2009).
[ShaRan et al.2003]	SHAIKHALI, A.; RANA, O.F.; AL-ALI, R.; WALKER, *D.W. UDDIe: An Extended Registry for Web Services. 2003,*

[ShiAda et al.2010] SHIH-HSI LIU; ADAM CARDENAS; XANG XIONG; MARJAN MERNIK; BARRETT R. BRYANT; JEFF GRAY; LIU, S.-H.; CARDENAS, A.; XIONG, X.; MERNIK, *M.; et al. A SOA Approach for Domain-Specific Language Implementation; Services, IEEE Congress on* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA*. proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 535–542 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.119].

[SinHuh et al.2005] SINGH, M.P.; HUHNS, M.N.; SINGH, M.P.; HUHNS, *M.N. Service-oriented computing: Semantics, processes, agents;* Wiley: Chichester, 2005 (ISBN 0470091487).

[SpaZis2010] SPANOUDAKIS, G.; ZISMAN, A. Discovering Services during Service-Based System Design Using UML. *IEEE Transactions on Software Engineering* **2010**, *36* pp. 371–389 [http://doi.ieeecomputersociety.org/10.1109/TSE.2009.88].

[SplBra et al.2009] SPLUNTER, S. VAN; BRAZIER, F.M.T.; PADGET, J.A.; RANA, O.F. Dynamic Service Reconfiguration and Enactment using an Open Matching Architecture. In *Proceedings of the 1st International Conference on Agents and Artificial Intelligence*: *Porto, Portugal, January 19 - 21, 2009*. Filipe, J., [Ed.], pp 533–539. INSTICC Press: Setúbal, 2009 [http://opus.bath.ac.uk/16178/].

[StAlJo2008] STEFAN DIETZE; ALESSIO GUGLIOTTA; JOHN DOMINGUE. Towards context-aware semantic web service discovery through conceptual situation spaces. In *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008),* pp 1–8. ACM Press: Beijing, China, 2008 [http://doi.acm.org/10.1145/1361482.1361488].

[SteSha1994] STERLING, L.; SHAPIRO, *E.Y. The Art of Prolog: Advanced Programming Techniques,* 2nd; MIT Press: Cambridge (USA), London, 1994 (ISBN 0-262-69163-9).

[StGrAb2007] STUDER, R., GRIMM, S., ABECKER, *A., Eds.: Semantic Web Services: Concepts, Technologies, and Applications;* Springer-Verlag GmbH; Springer: Berlin, Heidelberg, 2007,

[Syc2010] SYCARA, *K. Ontologies and Agents: NATO Advanced Study Institute on Software Agents, Sep 16-23, 2010.*

[SycWid et al.2002] SYCARA, K.; WIDOFF, S.; KLUSCH, M.; LU, J. Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems* **2002**, *5* (2) pp. 173–203 [http://dx.doi.org/10.1023/A:1014897210525].

[TalWor2008]	TALIS; WORD TRANSCRIPT.*Transcript: Sir Tim Berners-Lee Talks with Talis about the Semantic Web*, Talis; Word Transcript 20.02.2008 [http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html] (accessed 29. Mar. 2009).
[ThiKon et al.2009]	THIELE, A.; KONNERTH, T.; KAISER, S.; KEISER, J.; HIRSCH, *B. Applying JIAC V to Real World Problems: The MAMS Case.* In *Multiagent system technologies*: *7th German conference, MATES 2009, Hamburg, Germany, September 9 - 11, 2009 ; proceedings.* Braubach, L., Hoek, W. van der, Petta, P., Pokahr, A., van der Hoek, W., [Eds.]. 5774; Springer: Berlin, 2009, *5774,* pp 268–277 [http://dx.doi.org/10.1007/978-3-642-04143-3_29].
[ToDePe2009]	TOSI, D.; DENARO, G.; PEZZE, M. Towards autonomic service oriented applications. *International Journal of Autonomic Computing* **2009**, *1* (1) pp. 58–80 [10.1504/IJAC.2009.024500].
[TomFox2006]	TOMA, IOAN; FOXVOG, *Douglas.Non-Functional Properties in Web Services: WSMO Deliverable, D28.4 v0.1; WSMO Working Draft - October 25, 2006*, DERI 2006 [http://www.wsmo.org/TR/d28/d28.4/v0.1] (accessed 24. Jul. 2011).
[TrPrCo2003]	TRASTOUR, D.; PREIST, C.; COLEMAN, *D. Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches.* In *Proceedings*: *16 - 19 September 2003, Brisbane, Queensland, Australia.* IEEE Computer Society: Los Alamitos, Calif., 2003, p 222 [http://portal.acm.org/citation.cfm?id=942793.943143&coll=Portal&dl=GUIDE&CFID=59898292&CFTOKEN=52577230].
[TsaSha et al.2010]	TSAI, W.-T.; SHAO, Q.; SUN, X.; ELSTON, J.; ELSTON, *J. Real-Time Service-Oriented Cloud Computing.* In *6th World Congress on Services 2010*: *Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 473–478 [http://doi.ieeecomputersociety.org/10.1109/SERVICES.2010.127].
[Vas1998]	VASUDEVAN, *V. A Reference Model for Trader-Based Distributed System Architectures*, Object Services and Consulting Inc., 1998 [http://www.objs.com/survey/trader-reference-model.html].
[Vas2007]	VASILIEV, *Y. SOA and WS-BPEL: Composing Service-Oriented Architecture Solutions with PHP and Open-Source ActiveBPEL;* Packt Publishing: 2007 (ISBN 184719270X).
[VedOrc et al.2007]	VEDAMUTHU, ASIR S.; ORCHARD, DAVID; HIRSCH, FREDERICK; HONDO, MARYANN; YENDLURI, PRASAD; BOUBEZ, TOUFIC; YALÇINALP, *Ümit.Web Services Policy 1.5 - Framework: W3C Recommendation; 04 September 2007 [*http://www.w3.org/TR/ws-policy/] (accessed 18. Feb. 2011).

[VerGom et al.2005] VERMA, KUNAL; GOMADAM, KARTHIK; SHETH, AMIT P.; MILLER, JOHN A.; WU, *Zixin.LSDIS : METEOR-S: Applying Semantics in Annotation, Quality of Service, Discovery, Composition, Execution; Introduction*, LSDIS lab, University of Georgia 2005 [http://lsdis.cs.uga.edu/projects/meteor-s/] (accessed 30. Aug. 2010).

[VetLen2005] VETERE, G.; LENZERINI, M. Models for semantic interoperability in service-oriented architectures. *IBM Syst. J.* **2005**, *44* pp. 887–903 [http://portal.acm.org/citation.cfm?id=1126970.1126983&coll=Portal &dl=GUIDE&CFID=59898292&CFTOKEN=52577230].

[VoBeIa1995] VOGEL, A.; BEITZ, A.; IANELLA, *R. Discovery and Access of Services in Globally Distributed Systems.* DSTC Symposium; DSTC Pty. Ltd.: Brisbane, Australia, 1995,

[W3C2004] W3C.*WS Choreography Model Overview: W3C Working Draft 24 March 2004*, W3C 24.03.2004 [http://www.w3.org/TR/ws-chor-model/] (accessed 03. Dec. 2010).

[W3C2006] W3C.*Web and Service Oriented Architectures (1): Slideshow. 06.10.2006 [*http://www.w3.org/2006/Talks/1011-plh-soa/#(1)] (accessed 21. Mar. 2009).

[W3C2009a] W3C OWL WORKING GROUP.*OWL 2 Web Ontology Language Document Overview: W3C Recommendation 27 October 2009*, W3C OWL Working Group 27.10.2009 [http://www.w3.org/TR/owl2-overview/] (accessed 10. Aug. 2010).

[W3C2009b] W3C.*W3C Semantic Web Activity*, W3C 24.03.2009 [http://www.w3.org/2001/sw/] (accessed 29. Mar. 2009).

[WalArn2001] WALDO, J.; ARNOLD, *K. The Jini specifications*, 2. ed.; The Jini technology series; Addison-Wesley: Boston, Mass., 2001 (ISBN 0201726173).

[WanTsa et al.2010] WANG, M.-F.; TSAI, M.-F.; TANG, C.-H.; HU, J.-Y. Service Mining for Composite Service Discovery. In *Advances in Intelligent Information and Database Systems.* Nguyen, N., Katarzyniak, R., Chen, S.-M., [Eds.], *283,* pp 125–131. Studies in Computational Intelligence Springer Berlin / Heidelberg: 2010 [http://dx.doi.org/10.1007/978-3-642-12090-9_11].

[WaZhSu2004] WANG, H.; ZHANG, Y.-Q.; SUNDERRAMAN, *R. Soft Semantic Web services agent.* In *Fuzzy Information, NAFIPS '04: IEEE annual meeting of the North American Fuzzy Information Processing Society, Banff, Alberta, Canada ;.* fuzzy sets in the heart of the Canadian Rockies ; June 27 - 30, 2004. Dick, S., [Ed.]. IEEE Operations Center: Piscataway, NJ, 2004, *1,* pp 126–129 [doi:10.1109/NAFIPS.2004.1336263].

[WeeWar2010] WEERASINGHE, T.; WARREN, *I. Odin: Context-Aware Middleware for Mobile Services.* In *6th World Congress on Services 2010: Proceedings : SERVICES-1 : 5-10 July 2010, Miami, Florida, USA.* proceedings ; [including workshop papers]. IEEE; IEEE Computer Society: Piscataway, NJ, 2010, pp 661–666 [http://ieeexplore.ieee.org/ielx5/5575322/5575460/05575528.pdf?tp=&arnumber=5575528&isnumber=5575460].

[WeiBel2002] WEIBEL, N.; BELOTTI, *R. Web Services Technologies.* SOAP vs. Jini; Term Project; Swiss Federal Institute of Technology, 28. Jul. 2002 [http://www.rudibelotti.com/doc/projects/webservices/webservices.pdf].

[Woo2000] WOOLDRIDGE, M. Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* Weiss, G., [Ed.], pp 27–78. MIT Press: Cambridge, Massachusetts, 2000.

[WuRan et al.2007] WU, ZIXIN; RANABAHU, AJITH; GOMADAM, KARTHIK; SHETH, AMIT P.; MILLER, *John A.Automatic Semantic Web Services Composition,* LSDIS lab, University of Georgia 2007 [http://www.cs.uga.edu/~jam/papers/zLSDISpapers/zixin.doc] (accessed 12. May. 2009).

[Xia2007] XIAO, Q. A Language for Reliable Service Composition. *Lecture Notes in Computer Science (LNCS)* **2007** (No. 4362 (2007)) pp. 554–565 [http://dx.doi.org/10.1007/978-3-540-69507-3_48].

[Xia2008] XIAO, *X. Technical, commercial and regulatory challenges of QoS: An Internet service model perspective.* The Morgan Kaufmann series in networking; Elsevier/Morgan Kaufmann: Amsterdam, 2008 (ISBN 0123736935).

[YeChe2006] YE, L.; CHEN, *J. Automatic Composition of Semantic Web Services - A Theorem Proof Approach.* In *The Semantic Web (ASWC 2006): First Asian Semantic Web Conference, Beijing, China, September 3-7, 2006, Proceedings.* Mizoguchi, R., Shi, Z., Giunchiglia, F., [Eds.]. Lecture Notes in Computer Science, LNCS 4185; Springer; Springer-Verlag GmbH: Berlin Heidelberg, 2006, pp 481–487 [http://dx.doi.org/10.1007/11836025_46].

DEFINITIONS

EXAMPLES

INDEX