D4.3 Report on dynamically customizable services enhancing products

Afsarmanesh, H.; Sargolzaei, M.; Shafahi, M.

DOI
10.13140/RG.2.1.2124.5607
Publication date
2014
Document Version
Final published version

Grant Nº 285273

Glocal enterprise network
focusing on customer-centric collaboration

D4.3
Report on dynamically customizable services enhancing products

Edited by
UvA

May 2014

Project funded by the European Commission under the
ICT – Factories of the Future Programme

Contract Start Date: 1 Sep 2011 **Duration:** 42 months

GloNet WP4

CUSTOMIZED SERVICE-ENHANCED PRODUCT SPECIFICATION

Deliverable data

Deliverable no & name	**D4.3 – Report on dynamically customizable services enhancing complex products**
Main Contributors	*UvA – Hamideh Afsarmanesh, Mehdi Sargolzaei, Mohammad Shafahi*
Other Contributors	*UvA – Ozgul Unal*
Dissemination level	*Public*
Date	*May 2014*
Status	*Final* *Reviewed by: Luis M. Camarinha-Matos, Thomas Maltesen, Victor Thamburaj*

Deliverable summary

This deliverable, addresses the design of the proposed approach and the set of mechanisms for registration, discovery, recommended ranking, and composition of business service components related to each complex product. Therefore, it reports on the findings of the Task 4.3 in workpackage 4 (WP4) of the GloNet project.

The deliverable looks into details of business service specification and registration. It studies the functional/non-functional requirements of the planned so-called service specification tool (SST) and then identifies the needed meta-data for service specification. The introduced meta-data for service registration consists of 4 main components, namely: syntax, semantics, behaviour, and quality criteria, which are described in the deliverable. Then the process of discovery and ranking of services that match products is described. This deliverable also covers some enhancements to the PST Tool that is previously described in D4.1 and D4.2 and also addresses sub-products discovery and ranking.

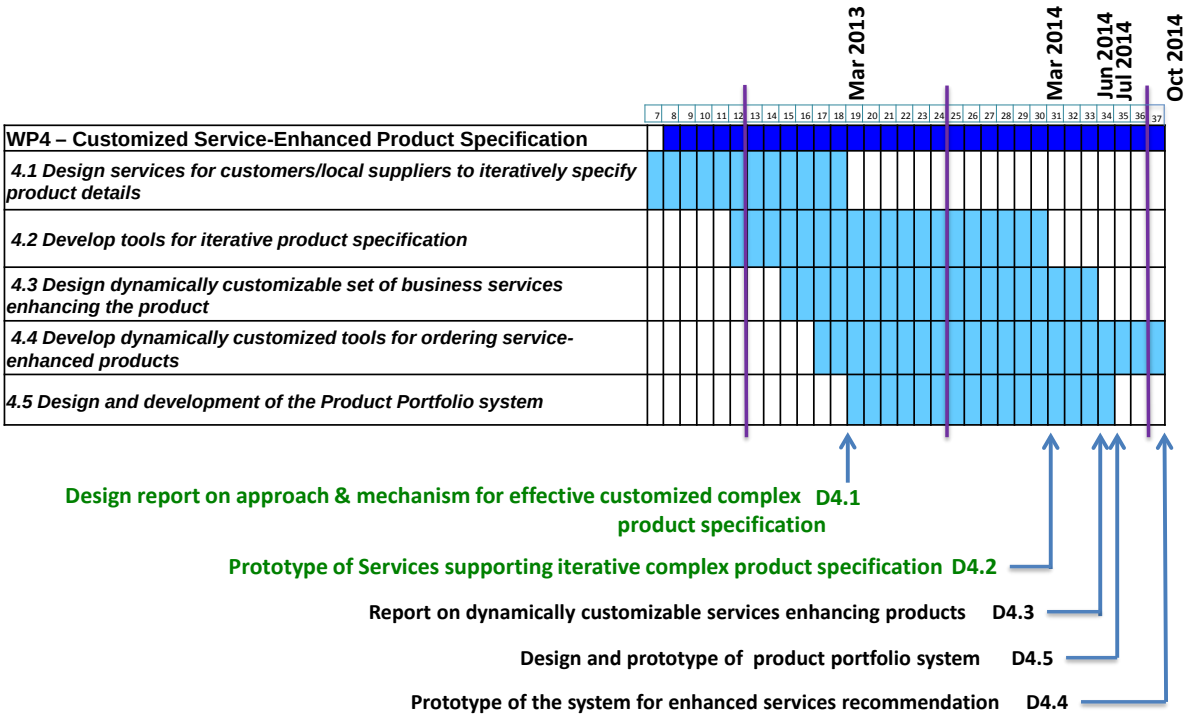
Finally, the deliverable discusses supporting of service-enhanced product recommendation and provides details about the recommendation approaches and the profiling which is preformed to tune the recommendations.

TABLE OF CONTENTS

PROJECT-RELATED SUMMARY

This deliverable is the third outcome of WP4. It presents a continuation of the previous two deliverables, acting as an enhancer to the conceptual designs of components addressed in D4.1. Being produced as the result of Task 4.3, it presents the second main step in complex product specification, namely corresponding to the design of dynamically customizable set of *business services* that enhance the complex products, while findings reported in deliverables: D1.1 ("Detailed requirements for GloNet use case and domain glossary"), D1.2 ("Specification of business scenarios"), D2.4 ("Mechanisms for defining composed services to support collaboration"), and D2.1 ("Required information/knowledge provision services specification") constitute the base for complex product specification. The approach addressed in D4.1 ("Design report on approach and mechanism for effective customized complex product specification") which was developed in D4.2 ("Prototype of Services supporting iterative complex product specification") constitute the main immediate inputs for this deliverable.

As shown in the figure that follows, the functionality, which is designed in this deliverable, will be implemented in WP4, through the task T4.4.



In D4.3, the design for product specification tool (the so called PST tool) described in D4.1 is further expanded to support product specification reusability. However, the main emphasis of D4.3 is on the introduction of the advanced service specification tool, called SST, for specification of business services related to the complex product, as well as further supporting their reusability and potential integration.

Together, the PST and SST tools effectively support the detailed complex product specification.

As such, deliverable D4.3 aims at the design of the an approach and mechanisms for service specification, registration, discovery, recommended ranking and composition of service components

for each complex product, together with the required underlying database model and the needed data manipulation functionality. Therefore, establishing a good design in this deliverable is imperative to the success of the final deliverables of WP4, specifically the deliverable D4.4, which is related to complete complex product specification.

Furthermore, the results generated by this sub-system developed in WP4 constitutes an important input both for VO formation process in WP5, as well as for generating input to be stored and manipulated in the product portfolio.

In the next step of WP4, during the task T4.4, the prototype of the system designed in D4.3, as well as its planned web services will be implemented. This development aims to enhance the developments reported in T4.2, to further support service registration and also to enhance the system by providing recommendations and suggestions of products and services needed by the Product/Service Discovery & Recommendation Engine (PSDR).

1 INTRODUCTION

Complex products (e.g. solar power plants and intelligent buildings) are one of a kind in their design specification and require careful customization of their production (e.g. tailored design of what exactly fits each case). Furthermore, the Product Life Cycle (PLC) of such complex products, as shown in Figure 1, runs over several decades, during which the products may need heavy maintenance, overhaul, and even evolve. The specification of complex products is therefore typically not performed in one session, rather iteratively, and potentially involving a number of different stakeholders, from equipment manufacturers and service providers, to experts at an EPC (Engineering, Procurement, and Construction) company. These stakeholders need to collaborate within a coopetitive[1] environment, in order to gradually and incrementally specify different components and sub-components related to the complex product.

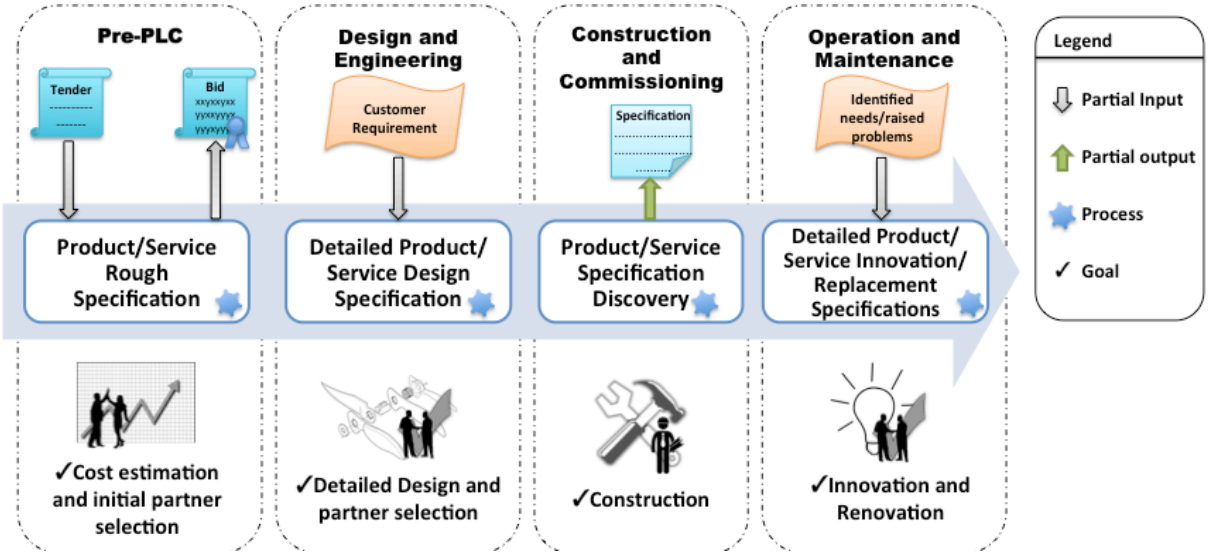


Figure 1. Service-enhanced product specification in different phases of complex products' PLC

Additionally, based on our findings in the two areas of solar plants and intelligent buildings, which are relatively young industries, the design and engineering of these complex products cannot be resulted through the mere searching and identification of the needed components among the existing products in the market. In other words, although familiarity with the existing related product/service details, as provided by different manufacturers and suppliers in the market, are the necessary starting point for the complex product designer, the mere existence of these product details is not sufficient to fully specify the ***Design*** of the complex product. Rather, the nature of

[1] Coopetition is the interaction that is with partial congruence of interests between organizations. In such an interaction, organizations cooperate with each other to reach a higher value creation, when compared to the value created without interaction, and struggle to achieve competitive advantage. Coopetition often takes place when organizations that are in the same market, e.g. within the same VBE, work together in the exploration of knowledge and research of new products/services. This can occur at the same time that they compete for market-share of their products/services and in the exploitation of the created knowledge. Not only two companies can interact within a coopetitive environment, but also several partnerships among competitors are possible.

these targeted complex products mandates detailed and concise design and customization processes for their components, including the equipment, or devices (here called sub-products), and their enhancing services, as well as involving different stakeholders in these processes. Please note that in this document the term service refers to business services related to a complex product.

In previous related documents [1] [2], we have proposed an environment for complex product specification, to support the so-called Coopetition needed through different PLC phases, as Figure 1 illustrates. In [2] investigated requirements for a product/service specification environment that enable further and more complex Coopetition, within the context of VBEs and their goal oriented Virtual Organizations (VOs). These functionalities are crucial when considering the coopetition environment that is supported in the complex product VBEs, a main aim in this environment is to support *modularity, sharing, and* the *reusability* of the generated assets as addressed in [1] and also later addressed in this deliverable.

The targeted complex products belong to young industries, and therefore their stakeholders can very much benefit from sharing assets such as specification of sub-products that are designed by others. Therefore, supporting both the reusability of sub-product specifications and the possibility of granting access privileges on them to other stakeholders are important requirements for this subsystem. Furthermore, considering that the targeted complex products are one-of-a-kind, their designed sub-products can be reused only in the case when sub-product's specifications follow a modular design approach, so that the pieces of their specification can be discovered within the VBE, accessed and copied for reuse. This constitutes another important requirement to be supported by this subsystem.

Considering the defined architecture for GloNet components, in this document we address the design of both the **Service Specification & Registration** and the **Product/Service Discovery & Recommendation**. These two subcomponents complete the development of *product/service specification* in this as illustrated in Figure2.

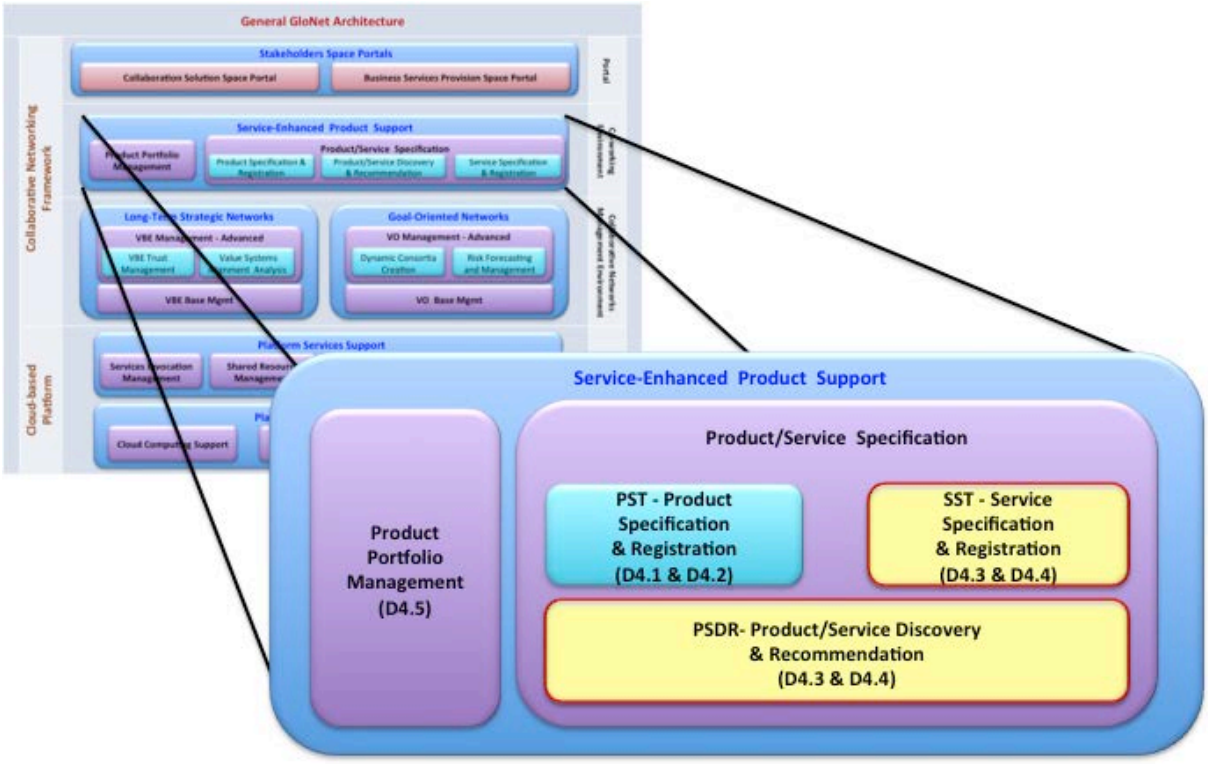


Figure 2. Service-Enhanced Product Support

Therefore, besides the specification of various equipment and devices needed for complex products, which are addressed before in [1] and [2], here we address the specification of the variety of needed business services. The implementation of these services range from *software systems* to *manual human-provided tasks*, which in one way or another enhance the complex product.

In the knowledge-based economy, services have an increasingly important role in manufacturing industries, which use functionality provided by services to differentiate their products [3]. In fact, by adding business services to complex products, while they increase the value of the products, a higher level of differentiation can be realized between different complex products [4]. Therefore, in our design of the specification framework for complex products, we consider that the design of sub-products typically include also design of a set of business services that offer some beneficial enhancement of these products. Capturing different aspects of these business services as well as the inter-relationships between these services and the sub-products of the complex product (e.g. devices), are addressed and supported by our proposed complex product specification framework and system. Many approaches and standards have been developed by the research community in the area of Service Oriented Architecture (SOA) that can be applied to specify and formalize business services [5]. There are however still challenging and open questions remaining related to making these services interoperable, so that they can be shared and reused. Another challenge is also how to assist authorized service providers with composing some existing services, thus producing value-added services to support complex products. Furthermore, there are still some gaps in correlation between services and sub-products in the context of complex products.

Other than considering service specification and registration, we should also bear in mind that *modularity, sharing, and reusability* of generated assets (being sub-products or services), which constitute some of the main functionalities of the product/service specification sub-system, cannot be accomplished when different stakeholders are not properly supported to discover such existing assets in the system. This is especially the case when considering that multitudinous (hundreds to thousands) components are present in each of the complex products and shall be dealt within their PLC. We approach this problem by providing mechanisms and functionalities for Product/Service Discovery and Ranked Recommendation that reduce the complexity and effort needed to specify, find, and reuse the existing specification of products and services.

The following sections of this deliverable provide more details about our design of these advanced functionalities related to service-enhanced product specification sub-system. These aspects are structured as follows in the next sections:

Section 2 – Business service specification and registration,
Section 3 - Service discovery and ranking of matched suggestions - service reusability
Section 4 – Sub-product discovery and ranking of matched suggestions - product reusability
Section 5 – Supporting service-enhanced product recommendation
then concluded with Section 6 on Concluding remarks, and Section 7 including references.

2 BUSINESS SERVICE SPECIFICATION AND REGISTRATION

Supporting business service specification and registeration/storage of these specifications in the context of complex products is an objective functionality in GloNet, which is addressed in this deliverable. In knowledge-based economy, services play important roles in the operation of industries and manufacturing firms [6], [7]. In GloNet, and in relation to complex products, we consider that each sub-product of a complex product when augmented with a set of supporting business services will offer beneficial value additions to both the customers and operation managers, related to that sub-product. Figure 3 shows an example monitoring business service (a software system in this case) for solar plants. A full description of such services, as well as more examples of business services in case of solar plants, which is the guiding use cases in GloNet, are defined in [8].

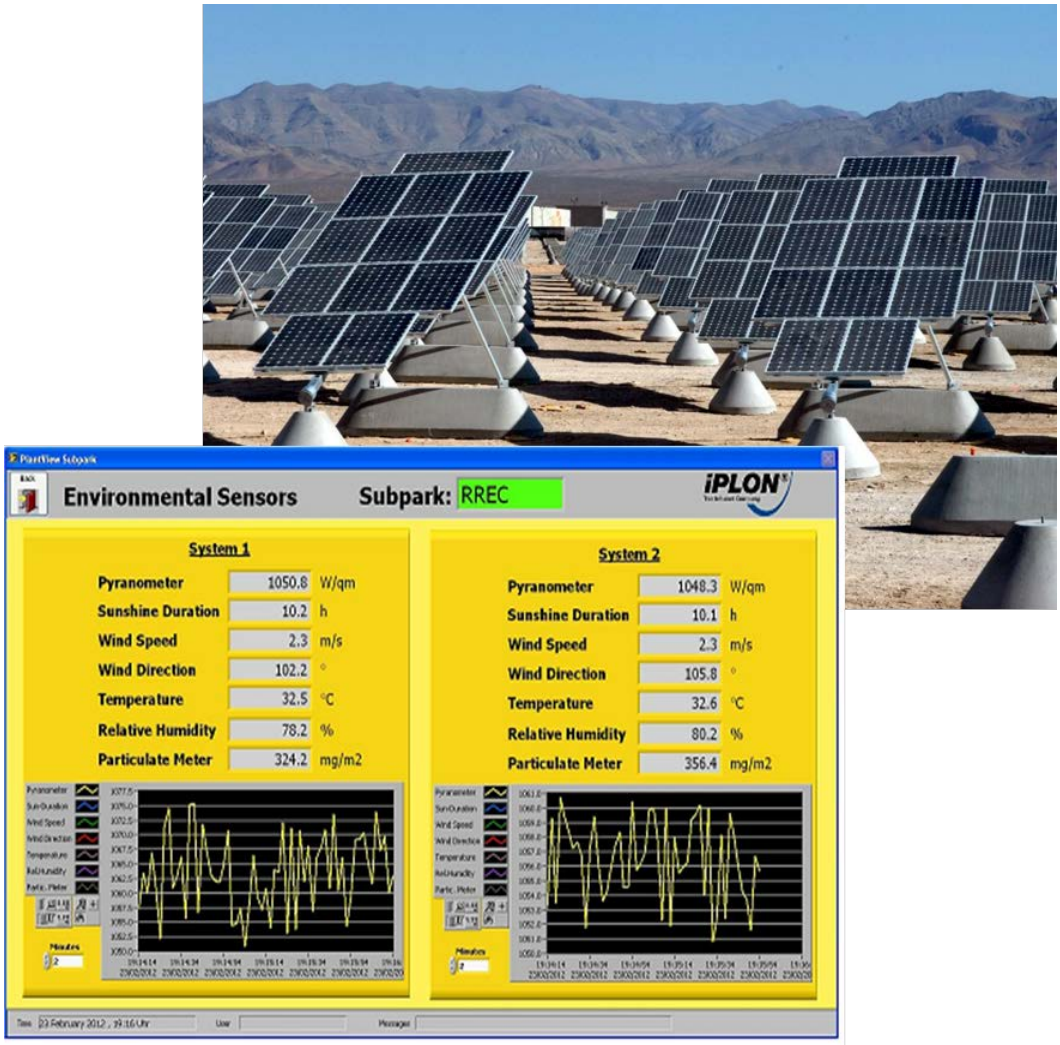


Figure 3- An example complex product enhanced by an atomic business service (materialized by a software service).

2.1 SST requirement and analysis

Similar to product specification, we also need to have *Service Specification* during different phases of the product life cycle (PLC) as shown in Figure 1. In fact, the first step to support service reusability

and interoperability is the specification of services. At the macroscopic level, the life cycle of complex products consists of three phases. Nevertheless, when it comes to the need for functionality provided by the Service Specification Tool (SST), the four stages mentioned earlier need to be considered, including the pre-PLC stage, as illustrated in Figure 4. The main emphasis for SST lies in three of these four stages as depicted inside red boxes. The need for each phase of the PLC is described briefly below.

- **Pre-PLC stage:** This stage deal with rough design specification, and main partner selection. In fact, it is related to the bid preparation stage of complex-products. In this stage, we need to identify services (manual task or software services) that are required to be added to the bid targeting the complex product, in response to the call for tender.
- **1st PLC phase- Design and Engineering phase:** This stage encompasses a fully detailed design, engineering, specification of and procurement of the complex products including their services and sub-products. Therefore, the design and Pre-engineering of the required services supporting and enhancing the complex products and sub-products should be done in this stage. Moreover, this stage includes evaluation/selection/extension of required services as related to complex product.
- **2nd PLC phase- Construction and Commissioning:** stage deals with the logistics, construction, and calibration of the complex products. Therefore, a number of functionalities need to be provided in this stage, to support the configuration and establishment of the needed complex product, as well as effective functioning of its constituting sub-products. In this stage, actors involved in constructing of complex products may need to discover, and retrieve detailed information about business services, for the purpose of service implementation.
- **3rd PLC phase- Operation and Maintenance**: It is considered as the long operation phase of an existing complex product. Therefore, it encompasses a large number of functionalities related to the operation, management and maintenance of the complex products and all its sub-products and services. The utilization of offered services including service discovery, composition and execution heavily continue in this stage, and will be used until the end of complex product's life cycle. Moreover, in this stage of PLC, service enhancement/replacement specification may occur. Finally, the innovation or design of new business services would be necessary for solving some emerged problems.

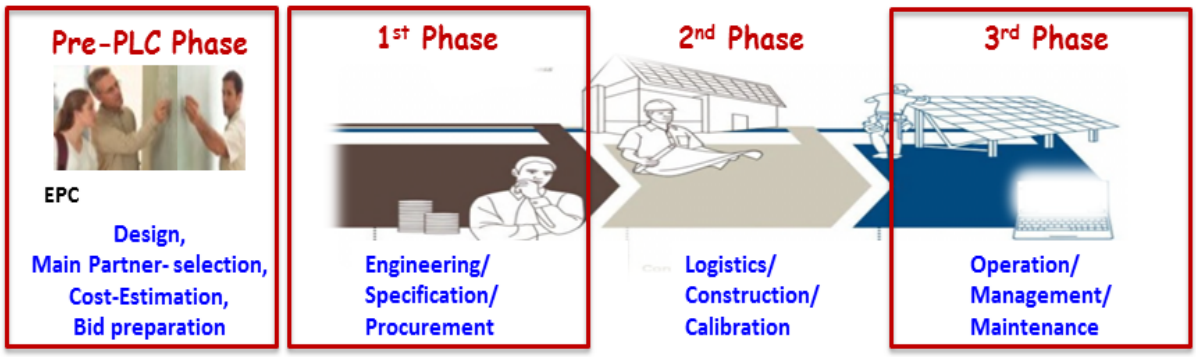


Figure 4: The product life cycle (PLC) phases

We have identified a set of functional and non-functional requirements for business services in the context of complex products as provided below:

The non-functional requirements mostly addressing the specifying needed meta-data for business services consist of the following:

i) *Formalization of service behavior* which tries to model the externally observable behavior of business services. This requirement will be meet in SST.
ii) *Unified Description* of Syntax, semantics and quality criteria of services which are also considered as a part of SST.
iii) *Sub-product/Service* interrelationships that aims to enhance sub-products with related services based on a relation between the specification of sub-products and provided meta-data for business services. This requirement will be implemented as a part of *Product/Service Discovery & recommendation* Module of *Service-Enhanced product Support* sub-system (see Figure 2).
iv) *Other non-functional requirements that have been previously identified and addressed as base requirements for the Product/Service Specification Sub-system (e.g. security and authorization, scalability and portability)[1][2].*

Besides the non-functional requirements, there are also some functional requirements for business services in GloNet as described below, which need to be supported:

i) *Service specification/registration* to store and index the specified meta-data for services. This is a part of SST module which is shown in Figure 2.
ii) *Effective service discovery* to search among registered services based on their specifications as a part of *Product/Service Discovery & recommendation* Module.
iii) *Support of Bundled/composite services* to make a bundle of atomic business services (software services or manual tasks) as an integrated composite business service. This bundling will be developed in SST.

Furthermore, service-enhanced products involve a number of different kinds of stakeholders/users, who are in one way or another using and/or benefitting from the SST tool. *Product/Service Suppliers and Providers* constitute the type of user for the SST tool, who are involved in the specification of real business services, in order to introduce/advertise these services for complex products development. The *Complex Product Customer* informs the EPC members/designers about the set of requirements for their needed business services to enhance sub-products. The EPC Members are the second type of users responsible to specify the needed services using the SST tool. These users are supported with the reuse of existing specified services in SST, for which they first discover existing services in SST that match the requirements presented by The *Complex Product Customers*. Finally, the *Monitoring Organization Staff* that constitutes SST users who monitor the execution of specified atomic and composite business services.

The remaining of this section focuses mainly on the *specification of business services*. Business services specifications are either provided by the GloNet partners who provide these services which are applicable to complex products, or by designers of these services. Business service specification primarily occurs during the first stage of the complex Product's life cycle. GloNet faces several challenges related to formalizing, sharing, and registering of the provided business services that enhance different sub-products and ultimately the complex products, as will be discussed in the coming sections. In this section we first introduce the topic of service specification for supporting complex products in GloNet. We then propose a novel approach for unambiguous formalism of the offered business services, such that it can later support their semi-automated discovery and ranking, and thus assist with their reusability and potential composition of integrated services. We address support of service discovery as a part of *Service-Enhanced product Support* sub-system, in Section 3.

2.2 Business Services meta-data

Rooted in a traditional definition for good and services [9] we describe a business service as follows:

"A business service causes a change or enhancement in the condition of a person, or a good belonging to some economic entity, brought about as the result of some task or activity performed by another economic entity, with the approval of the first person or economic entity". Similar to sub-products, services also have distinguishable ontological definitions through which they can be specified. Services are intangible, interactive, represent simultaneity in their production (execution) and consumption, and can be bound to a particular time (i.e. the time during which they can be delivered and executed) and place (i.e. the place that they can be delivered) [10].

Each business service has a delivery procedure, which is initiated by a triggering event. For example, the triggering event for the *Monitoring Software Service* (in Figure 3) can be the detection of an alarm, or a scheduling event that indicates periodic monitoring.

Also every business service represents the execution of certain actions in a certain order and manner [8], which implies the process describing the service. As Figure 5 shows, a business service can be realized through alternative kinds of business processes, and planned triggering events. Indeed, the notion of business service is an abstract construct that basically encapsulates its external view, and specifies what would be delivered through this service to the users.

Furthermore, a new business service can be realized through composition of several existing business services. As such, a composed business service represents the interaction and behavior of its component associated business services , and reveals how the composed service would be performed through them, and with which corresponding triggering events. Actions involved in the business service execution can be materialized either automatically, either through some software function (the so-called software services), manually through some human-based tasks (the so-called manual tasks), or through a combination of the mentioned two kinds. In a nutshell, we can mention that each Business Service (BS) may involve Business Processes (BPs), while each BP may involve either the invocation of granular software services or performance of manual tasks.

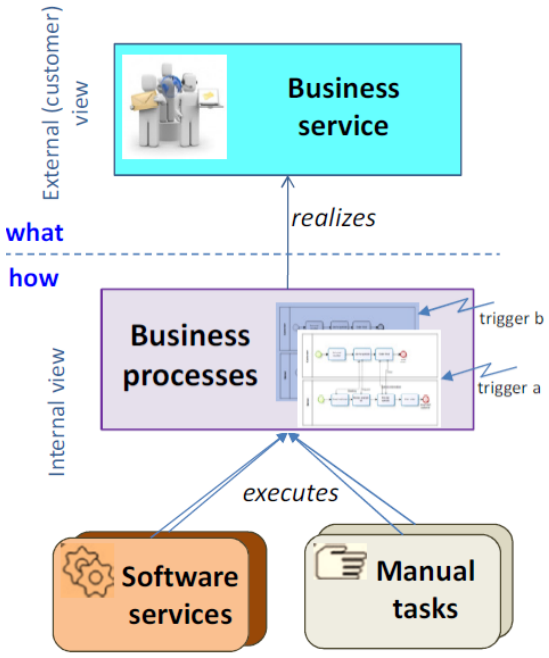


Figure 5: Views of business services

2.3 Business Service Specification

We formalize the definition of a business service (also called a *service* for short, in the remaining of this section) as a tuple *S = < Sn, Sm, Be, Qcs >,* where *Sn, Sm, Be, Qcs* respectively represent the *Syntax, Semantics, Behavior* and *Quality Criteria of Service.* These four aspects of this proposed service specification are further described below. Moreover, a number of notational options are adopted and applied for better representation of each of these four elements. Although the state of the art on service specification addresses the services' syntax, semantics, and Quality Criteria [5], we have further added service behavior in our design of the meta-data for business services. As such we introduce the following notation for formal representation of services, where each of these four elements, addressed in details.

- **Syntax**: Typically, the syntactic properties of a service are represented using the notation of ***XML***-based standards and languages, such as those provided by the web service description language (WSDL) [11] and Simple Object Access Protocol (SOAP) [12]. Some examples of syntactic aspects of the BS specification include: service *name*, the name of its *included operations,* and their *input* and *output* arguments, etc.

- **Semantics**: Conceptual properties of services, here referred to as semantics, are typically defined with an ***ontology*** notation, as an explicit specification of the conceptualization of the knowledge about services. The service ontology definition encompasses a group of vocabularies that specify semantic attributes of services (e.g. goal and category) and their inter-relationships, which together form a meaningful concept about each service. In fact, the semantic description of business services would enrich the information about services to the level that cannot be specified by their mere syntactic description. Some examples of the semantic aspects of the BS specification include *process description* of the *associated business processes* to the BS, the *textual description* and purpose-classification of the BS *goals* and *context*, *pre-conditions* and *post-conditions* of the BS. Moreover, service-class as a part of the semantics, aims to categorize services in order to match with product/sub-product during service enhanced recommendation. This is described in details in Section 5.

- **Behavior**: Besides semantic and syntactic descriptions of the services, we also need to specify and formalize the *externally observable behavior* of each service, which shall in turn represent the proper invocation order for its operations. These behavioral properties can be used later for the purpose of service discovery and integration functions. They will be used to improve the accuracy of service matchmaking to requirements [13]. The behavioral aspects of the BS specification address its functionality, based on which it can be unambiguously implemented by the software developers. We propose to formalize the behavior of services applying the ***Constraint Automata*** [7], where every *state* of a Constraint Automaton (CA) represents the externally observable internal configuration of a service, and every *transition* represents the exchange of one or more messages by this service.

  Please note that this aspect of service specification is useful for those services that are supported by software functions (software services), but we also have the behavioral specification for manual tasks because we have considered a simple web service for each manual task to show the start and end points of the task.

In fact, we use the CA notation to allow the user to capture the behavioral specification of a service by a finite number of states and some labelled transitions. The CA-based behavioral definition of a service enables software developers to follow the sequence of planned operations in order to decide and implement the behaviors of the service.

This behavioral specification also comprises essential information for automated service invocation in the case of stateful services [14]. Stateful services are defined where a client intends to keep some data or states during one invocation of the service, and then deploying those data and states during a subsequent invocation. In other words, the invocation of a stateful web service depends on its previous invocations. Briefly, the formal specification of the stateful services' behavior provided by Constraint Automaton specify the desired sequence for operations' invocation. This specification for stateless services would be several single states CA (one Constraint Automaton for each service's operation).

- **Quality Criteria of Service (QCS)**: While the service discovery is usually done according to the functional properties of the BS specification (i.e. syntax, semantics and behavior of services), non-functional properties of services, i.e. Quality of Service (QoS) parameters have also an important role in user's service selection. Therefore, we have specified some QoS metrics as quality criteria of services to assist users in service selection and improve the accuracy and optimization in service matchmaking. The QoS values of services are usually claimed by service providers and ensured through a service level agreement (SLA) as a part of a contract between the provider and the users [15]. Moreover, the users using a service may evaluate it by giving rates to the quality metrics as their feedback. In GloNet, we consider just the QoS values, which are measured and collected by the provider, but we would also considerthe calculated trustworthiness of the service provider [16] as a factor to estimate the final value of QCS properties. We have identified some quality criteria of assessment of offered services such as Execution duration, maximum response time, and availability, which are already defined in [8].

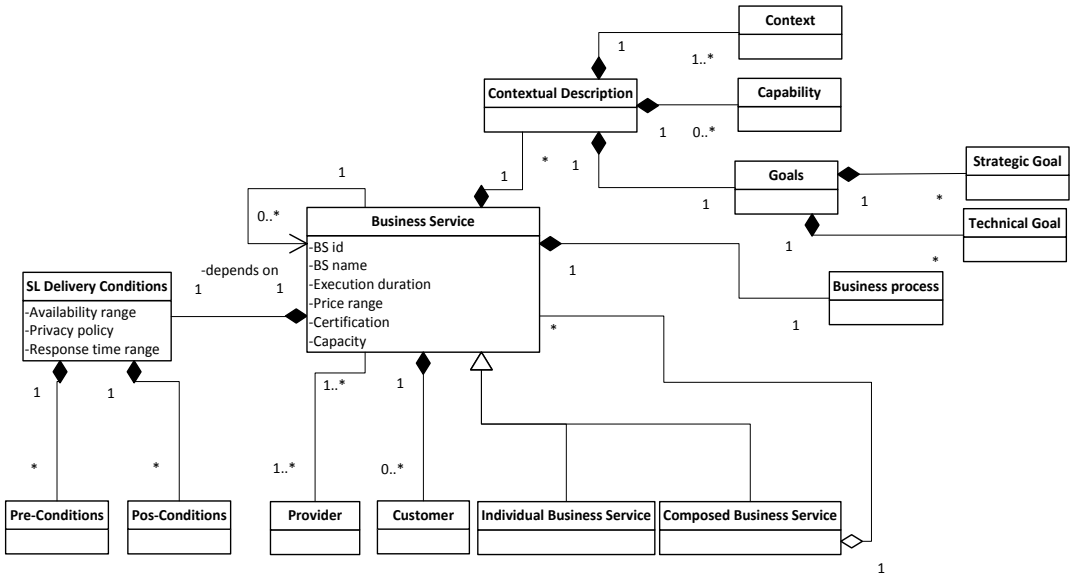


Figure 6. The business service model in UML notation

2.3.1 Business Services Classification

To support service-enhanced products, we need to discover and link relevant services to the specified sub-products or the complex product as a whole. Therefore, the proposed SST should consider a kind of classification for business services according to their utilities and application. For example, the business services of "Site Maintenance Services" and "Wildlife prevention" are relevant and can be assigned to the *Solar Plant* category, while "Check and Report" service for devices can be defined for in *Intelligent lamp* class.

As mentioned in the previous subsection, the *service-class* as a feature of semantics descriptions of business services implies a specific classification for services in the system. We introduce service-class as a basic class defined to model the generic categorization of all business services, where each *service-class* represents a specific set of services that are potentially belonging to the same class of services, and used in similar use cases.

Service classification is similar to the kind of classification for sub-products, which are represented and stored as object-classes [1].This classification can later be used to match a set of services based on their service-classes, to a set of sub-products, according to the sub-products object-classes. This matchmaking constitutes the base mechanism for recommendation of a set of related services that can potentially enhance the corresponding sub-product. In other words, service-classes can be used to identify and filter what services are relevant to which product/sub-products. For example, the service "Wildlife prevention" might be interesting for the customer ordering a solar panel, but it might not be interesting for customers who want to order for example an "*Intelligent lamp".*

Further to the above, introduction of service-classes in our service registration database model provides the following specific benefits for service definitions:

1. Flexibility to categorize the same services in different kinds of service-classes.
2. Guaranteeing that the user will provide the required (obligatory) input for the service-class of each service (e.g. the services classified as "atomic service" require to have their WSDL file defined).

We use the NACE ("Statistical Classification of ***Economic Activities*** in the European Community") [17] coding system for coding the classification of services. We have chosen the NACE Rev. 2 coding system because as its description suggests, this coding system can represent **economical activities** performed in industry. Being a standard coding system in the EU, this coding system also brings other benefits that are described in more details in section 4.1.3 and section 5.1.

2.4 Service Registration

The registration function is designed to register and store services according to their proposed specification. As the base, we have used the data structures designed for *product registration* in deliverable 4.1 and 4.2, and extended those structures for the *service registration*. Therefore, each element of the service specification would be registered as a data object, with four properties, including service's *Feature-Kind, Type, Value* and *Unit*. Please note that these four properties represented in the designed service registration are already defined in deliverable D2.4. Figure 6 shows the business service profile in the UML notation, which is addressed in D2.4. Figure 7, shows an example of a software service registration data (for *Check & Report Service*) according to the

proposed service specification of Figure 6, and the business service specification presented in Section 2.2. As you see in this figure, the registered properties are categorized into four **service specification aspects**, i.e. the syntax, semantics, behavior and quality criteria of services.

	Feature-Kind	Type	Value	Unit
	ID	int	241	-
Quality Criteria	Execution duration	int	60	Second
	Minimum-Price	int	10	$
	Maximum-Price	int	10	$
	Availability range	int	24	Hour/day
	Maximum Response time	int	100	day
	Name	String	Check & Report	-
Semantics	Context	String	Energy Consumption	-
	Technical Goal	String	Report provision	-
	Strategic Goal	String	Fault detection	-
	Capabilities	String	-	-
	Capacity	int	-	-
	Process Description	File	(Pointer to file)	BPMN
	Privacy policy	String	-	-
	Pre-conditions	String	-	-
	Post-Conditions	String	Confirmation	
Syntax	Syntax	File	(Pointer to file)	WSDL
Behavior	Behavior	File	(Pointer to file)	WSBS

Figure 7: Example of a "Software Service" Specification (Check & Report service– that checks the device's status and makes a report)

The data stored for registration of the manual tasks is also similar to the software services as exemplified in Figure 8. As mentioned earlier, for the purpose of monitoring of service executions, we also define a simple web service for each manual task that includes two basic operations: Start and Stop. Figure 8 shows the example of service registration for the manual tasks called *Wildlife Prevention,* related to the case of *solar plants*.

	Feature-Kind	Type	Value	Unit
	ID	int	172	-
Quality Criteria	Execution duration	int	5	day
	Minimum-Price	int	1000	$
	Maximum-Price	int	10000	$
	Availability range	int	10	Hour/day
	Maximum Response time	int	10	day
	Name	String	Wildlife Prevention	-
Semantics	Context	String	Maintenance	-
	Technical Goal	String	Cleaning of colonies of wildlife (rats, rabbits, bird nests, etc.)	-
	Strategic Goal	String	Deploy prevention	-
	Capabilities	String	-	-
	Capacity	int	10	km2
	Process Description	File	(Pointer to file)	BPMN
	Privacy policy	String	-	-
	Pre-conditions	String	-	-
	Post-Conditions	String	Confirmation	
Syntax	Syntax	File	(Pointer to file)	WSDL
Behavior	Behavior	File	(Pointer to file)	WSBS

Figure 8: Example of Service Specification for a "manual task" (Wildlife Prevention – Cleaning vegetation and applying prevention)

As mentioned in Section 3.3, we formalize the behavior of software services in terms of constraint automata. It is therefore necessary to generate a constraint automaton for each such service, which would be then captured and stored within the registry of services. The data structure for this registry consists of a state table, which stores: the current state, the next state, the operation names, and the data constraints of the constraint automaton [7]. Within the data object (e.g. in Figure 7) allocated to a service registration, in its behavior field, we just store a pointer to a file, which points to the corresponding behavior specification. Figure 9 shows the behavior specification of the *Check & Report Service*, which is registered in Figure 7, in terms of a constraint automaton. The formal notation used for this is called WSBS (Web Service Behavior Specification). The behavior registry is needed for services to serve two main purposes: (1) the need to match behavioral aspects of services, for the purpose of service discovery, and (2) to assist software service developers with unambiguously generating final executable code for every integrated service.

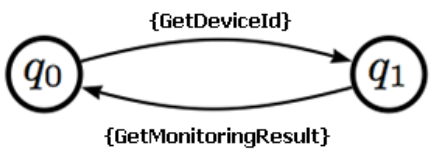


Figure 9: External Behavior Specification of the *example Software Service*

For the syntax specification, we also store a pointer to a file, which shows the corresponding WSDL [11] document of the registered service. For example, Figure 10 shows the WSDL document, which is registered as the syntax property for the *Check & Report Service.*

As you see in Figure 6, the data about semantics and QCS properties of a service would be registered in details as string type, therefore we have two databases for semantic and QCS data about services, as presented in Figure 15. The syntax and behavior are also designed to be stored in separate databases (see Figure 15). This design approach enforces the discovery functionality of the SST.

```
<definitions name="Monitoring"
  targetNamespace="http://namespaces.Monitoring-info.com"
  xmlns:es="http://www.Monitoring-info.com/Monitoring.wsdl"
  xmlns:esxsd="http://schemas.Monitoring-info.com/Monitoring.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetDeviceIdRequest">
    <part name="body" element="esxsd:GetDeviceId"/>
  </message>

  <message name="GetDeviceIdResponse">
    <part name="body" element="esxsd:GetDeviceIdResponse"/>
  </message>

  <portType name="GetDeviceIdPortType">
    <operation name="GetDeviceId">
      <input message="es:GetDeviceIdRequest"/>
      <output message="es:GetDeviceIdResponse"/>
      <fault message="es:GetDeviceIdFault"/>
    </operation>
  </portType>

  <binding name="MonitoringSoapBinding"
           type="es:GetDeviceIdPortType">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDeviceId">
      <soap:operation
        soapAction="http://www.Monitoring-info.com/Monitoring"/>
      <input>
        <soap:body use="literal"
          namespace="http://schemas.Monitoring-info.com/Monitoring.xsd"/>
      </input>
      <output>
        <soap:body use="literal"
          namespace="http://schemas.Monitoring-info.com/Monitoring.xsd"/>
      </output>
      <fault>
        <soap:body use="literal"
          namespace="http://schemas.Monitoring-info.com/Monitoring.xsd"/>
      </fault>
```

Figure 10: Syntactic Specification of the *Check & Report Service*

Figure 11 shows the design of an example user interface, which is designed to register an atomic business service according to our proposed service specification. Please note that the yellow icons field beside some fields (e.g. process description) indicates that the corresponding value for that should be uploaded from an external file. Once a business service is specified and registered, an ID for that business service is returned to the user, e.g. registering *Check & Report* service, will return the ID of 241 back to the user.

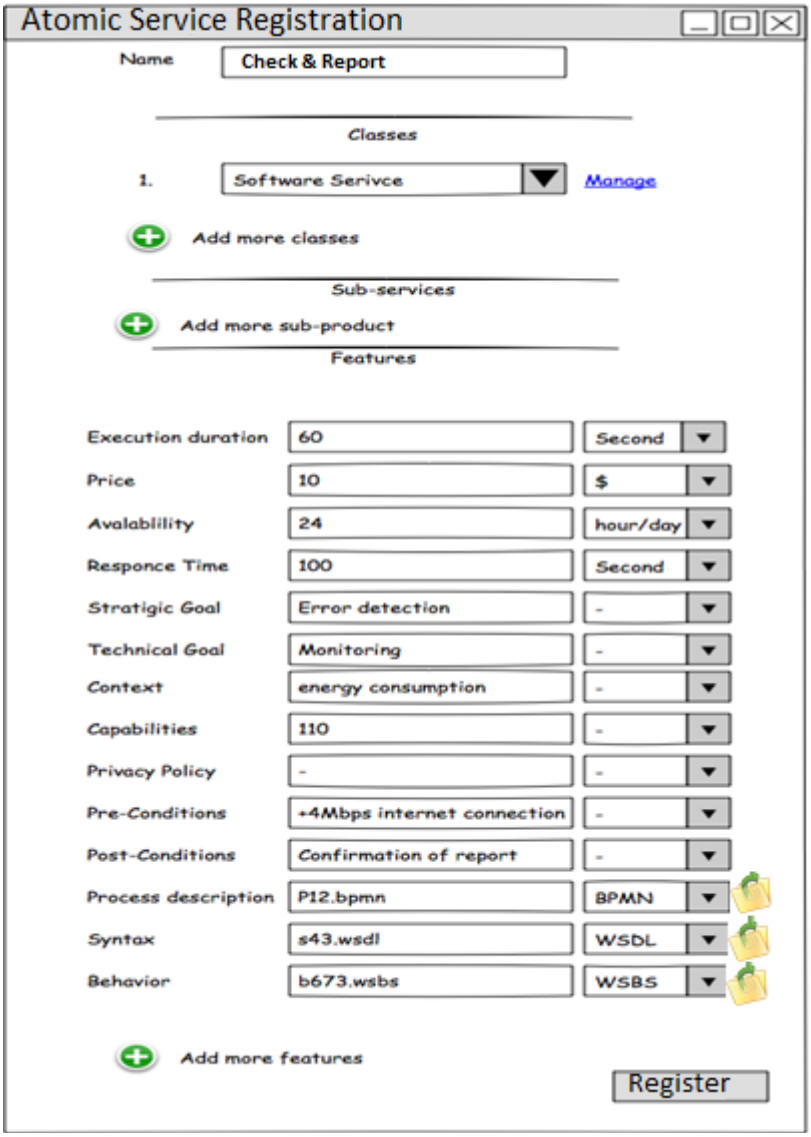


Figure 11: Example interface to register an atomic business service

As Figure 6 shows, it is possible to make a bundle of atomic business services, where their component **services** may include both software services and manual tasks to define a composed business service. We call this kind of services as **composite business service** in the sequence. Figure 12 shows an example of a composite business service, which is a site maintenance service for solar plants (see D2.4 for details). This composite business service consists of four atomic business services as its components, including Check & Report, Vegetation Management, Wildlife Prevention, and Water Drainage. As you can see in the figure, these four component business services are provided by three different companies, including: *Security Company*, *Site Cleaning Company* and *Wildlife Prevention Company* and the atomic services as the components can be a combination of software services and manual tasks.

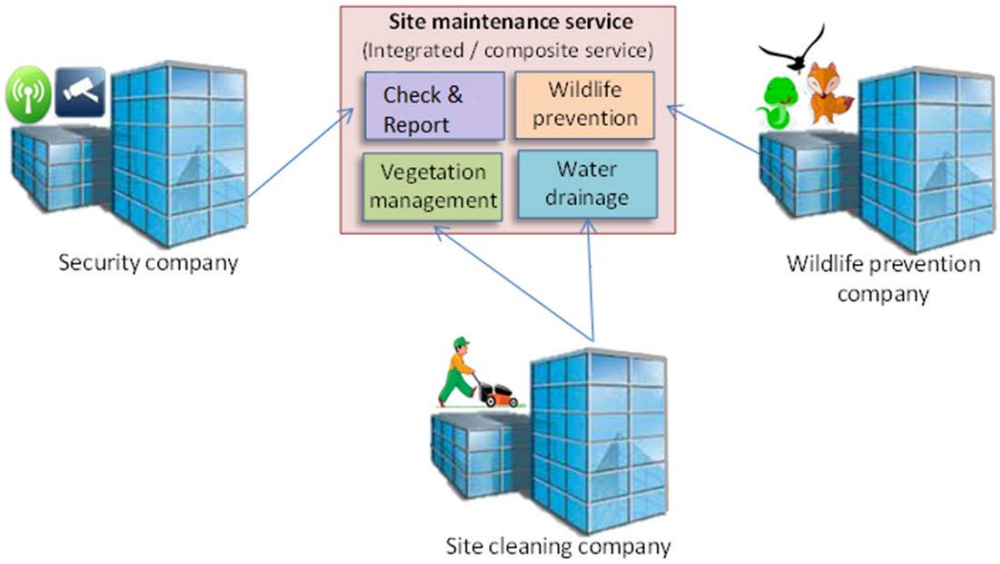


Figure 12-An example of composite business service: solar plant - site maintenance service

Figure 13 shows an example of service registration data for a composite business service that is depicted in Figure 12. As you see in Figure 13, of the information required for composite business service registration is similar to that for registration of atomic business services, except that it does not consist of syntax and behavior aspects since these are captured in details within their corresponding atomic BS registration. But additionally, the composite BS registration includes an aspect called *Bundling*, which consists of a feature-kind called *constituent* to specify the set of IDs for its component service constituents, e.g. 172 for *Check & Report*. Therefore, composite services are defined through their three aspects of Semantics, Quality Criteria, and Bundling. Please also note that the IDs mentioned in the constituent field represent atomic services, which are already specified and existing in the SST.

	Feature-Kind	Type	Value	Unit
	ID	int	272	-
Quality Criteria	Execution duration	int	15	day
	Minimum-Price	int	1500	$
	Maximum-Price	int	40000	$
	Availability range	int	10	Hour/day
	Maximum Response time	int	5	day
	Name	String	site maintenance	-
Semantics	Context	String	Maintenance	-
	Technical Goal	String	Removing natural caused damages	-
	Strategic Goal	String	Improve performance	-
	Capabilities	String	-	-
	Capacity	String	10	km2
	Process Description	File	(Pointer to file)	BPMN
	Privacy policy	String	-	-
	Pre-conditions	String	+10 Mbp Internet connection	-
	Post-Conditions	String	Confirmation	
Bundle of	**Constituents**	Service ID	241, 172, 567, 109	-

Figure 13: Example of a "composite Service Specification" (Site Maintenance business service)

Figure 14 shows the BPMN diagram, which is designed as the process description indicating the workflow of constituent services for the example composite business service specified in Figure 13.

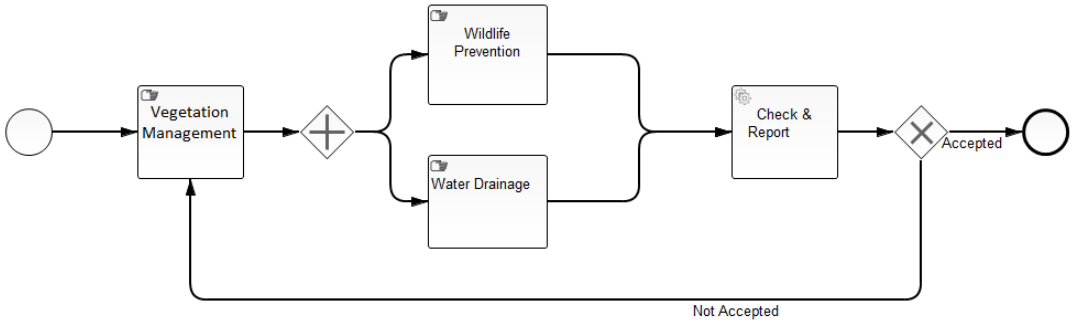


Figure 14: Example workflow for a composite business service (Site Maintenance business service)

Finally, Figure 15 shows the design of an example user interface, which is designed to register a composite business service in SST, according to our proposed service specification frame.

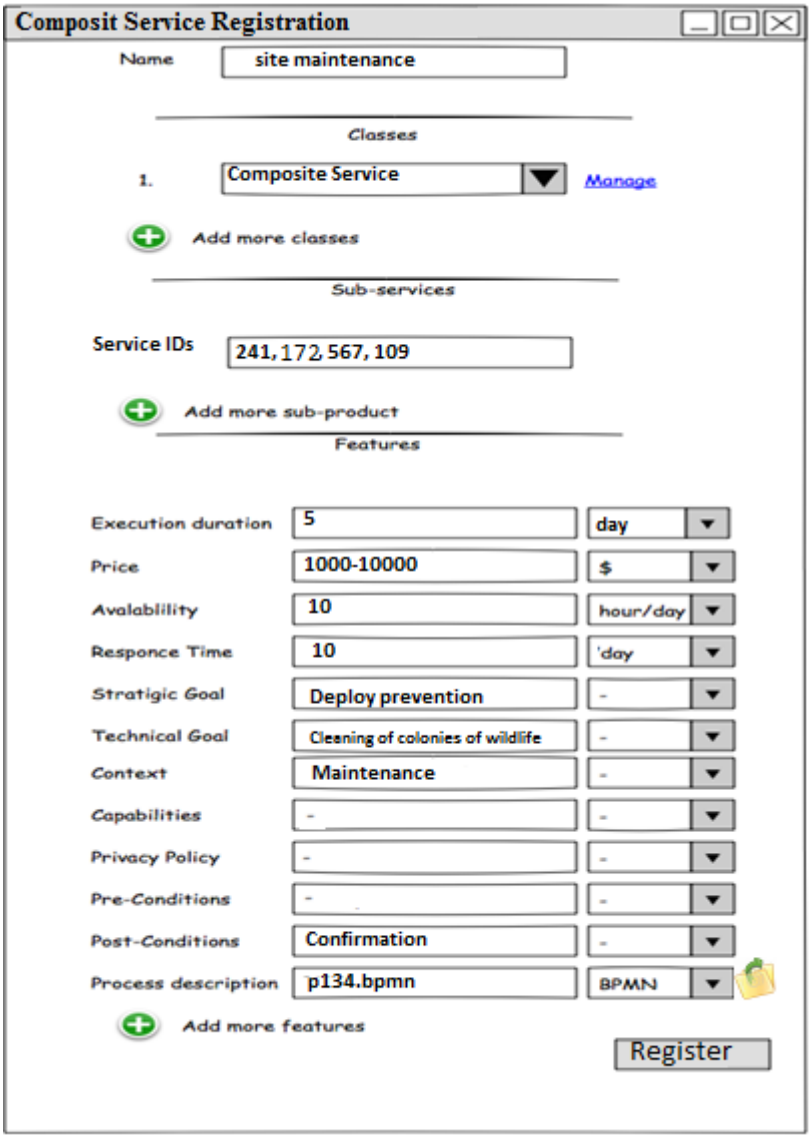


Figure 15: Example interface to register a composite service (Site Maintenance business service)

3 SERVICE DISCOVERY AND RANKING OF MATCHED SUGGESTIONS - SERVICE REUSABILITY

As we mentioned in the previous section, as a part of PSDR engine, we aim providing a tool for supporting service specification for complex products in the GloNet. Besides service specification, there are also several other challenges, related to service reusability , including the discovery, matching and suggesting of the most-fit registered business services for service reusability. In this section we therefore introduce the topic of service discovery, and propose a model to consider all specified aspects of the services during the discovery process. This section focuses on discovery of services offered by GloNet partners as service providers, which belongs to the 3rd step of the complex PLC: *Operation/ Management/ Maintenance* (see Figure 4).

As an example, suppose that there is an *EPC Member* who is an SST user and wishes to find a business service for a certain defined sub-product, while the service should satisfy the requirements set by the corresponding *Complex Product Customer*. As such, this business service will enhance that sub-product of the complex product. Furthermore, assume that several related business services are already implemented and provided by different partners within the VBE of the complex product, and further specified by the *Product/Service Suppliers and Providers* in SST. The *EPC member* can then discover such services matching the requirements set by the *Complex Product Customer,* and SST can assist this user with retrieving the best matched services among all existing business services. Therefore, SST automates the discovery of business services, serving as the base for identification of the most-fit partners to offer them. There are multiple approaches developed by the research community in the area of services discovery to match *Complex Product Customer*' requirements against descriptions of the existing services, but only a few earlier research addresses all aspects of service specification needed for efficient service discovery [5]. We want to go beyond the existing approaches by considering all four service aspects specified in the service specification including syntax, semantics, behavior and Quality Criteria of Service (QCS). In the next Sections, we introduce our proposed service discovery approach for GloNet.

3.1 SERVICE DISCOVERY APPROACH

As mentioned in section 2, two classless of materialization of business services can be defined for complex products, including the manual tasks and the software services. To have a uniform approach and functionality for service discovery as well as for supporting the other tools that monitor service executions, we also introduce a simple web service for each manual task, including two basic operations of: Start and Stop, which are invocated at the beginning and at the end of the manual task, that corresponds to the check-in & check-out of the manual worker providing that service. In other words, in terms of implementation, this allows to see all implementations in a uniform way as web services, which also facilitates the execution monitoring of business services.

This section addresses the *Service Discovery and Ranking of matched suggestions, which constitutes a part of the product / service discovery and recommendation module* of the general GloNet architecture (see Figure 2). It addresses design of mechanisms for discovering and matchmaking between the required criteria and the existing service specifications, to support service designers with offering the best-matched business services. The ranking is done according to the similarity score that expresses affinities between specification of each service and the users-submitted query. In other words, the discovery of most-fitting services can at best perform a search/match based on the service specification, including: (1) the syntactic interface for the services (e.g. specified as service names and operation names), (2) semantics related to the concepts and functionality of the

requested service (e.g. specified in terms of preconditions, assumptions, post-conditions), (3) behavioral properties (i.e. the desired sequence of operations' invocations) and (4) *QCS* and non-functional values of the services (e.g. specified execution duration, price and availability).

Conceptually, our service discovery proceeds in the following four successive steps:

I. Extract the needed meta-data from the *Service Specification and Registrations* for each registered service.
II. Process multi-faceted queries that represent user preferences for all service aspects.
III. Model an approximate bi-simulation [18] between a query and our services as a Constraint Satisfaction Problem (CSP).
IV. Solve the modeled problem to recommend the most-fit service(s), among the potential set of registered candidate services, and to rank the resulted services.

Step I is needed to retrieve the specification of registered services in order to compare and evaluate them during service matchmaking. For this step, we access the four registries designed to capture syntax, semantics, behavior, and QCS properties of the services (see Figure 16). As we mentioned in Section 2, we use WSDL documents as the syntax of services, and a kind of automata (Constraint Automata) in the so-called WSBS notation as the behavioral specification for services. We will create the WSBS files using the WSDL and some extra necessary annotations. The semantic description of services can be represented using a language like OWL-S, and finally QCS values can be captured in a traditional record. In *step II*, we obtain a *multi-faceted query* from the user, containing values and constraints for the syntax, semantics, behavior and QCS, which address user's preferences. We process the query to find similarities between user's request and actual services in the database. In *step III*, we set up a CSP, where soft-constraint functions are assembled using the similarity scores derived in step ii. These similarity scores are measured based on *syntax, semantic* and *QCS* properties of the services. At the same time, we define those constraints that compare the two *behavioral* properties of the *multi-facted query* and each registered service, and measure their similarities. Finally, we find the best solutions for this *CSP*, and we return them to the user. These steps can be implemented by different software modules whose global architecture is defined in Figure 16.

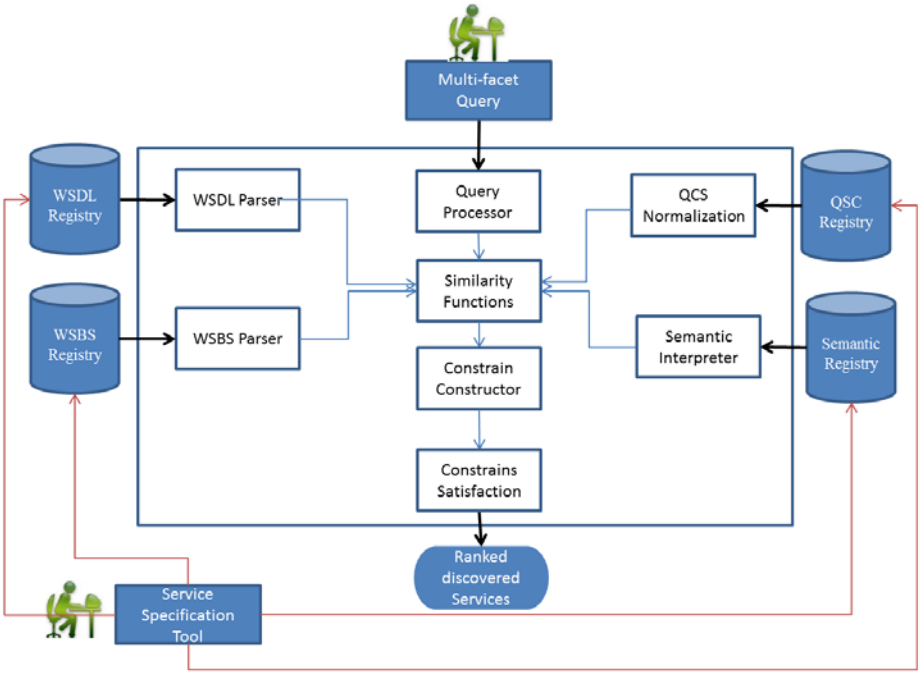


Figure 16. The designed architecture for service discovery and recommendation

3.2 Planned Architecture

Figure 16 shows the planned architecture for service discovery in GloNet, which consists of 9 software modules. The description of these modules is as follows.

- **Service Specification Tool.** This tool specifies existing services according the proposed service specification (see Section 2.2). Moreover, this tool registers the results of the specification as meta-data in the corresponding registry. We have designed four registries for different aspects of the service specification including *WSDL Registry* (for syntax), *WSBS Registry* (for behavior), *Semantic Registry* (for semantics) and QCS Registry (for quality criteria of services).

- **WSBS Parser.** While a WSDL document specifies the syntax and the technical details of a service interface, it lacks the information needed to convey its behavioral aspects. In fact, a WSDL document only reveals the operation names and the names and data types of their arguments; it does not indicate the permissible operation sequences of a service. If we know that a WS is stateless, then all of its operations are permissible in any order. For a stateful service, however, we need to know which of its operations is (not) allowed in each of its states. In [14], some of the authors of this paper have already formalized the behavior of a WS (i.e., the WSBS) in terms of CA. All specified WSBSs are stored in a WSBS Registry (see Figure 16). We can automatically extract a single-state automaton from the operations defined in a WSDL document describing a stateless WS. For stateful WSs, we can define a state table as a WSBS, or develop an interactive tool that (using a GUI) allows a programmer to visually create the automaton states describing the behavior of a service, and tag its transitions with the operations defined in its WSDL document. Moreover, before estimating the approximate bi-simulation between a specific service and a query, the WSBS document of the service should be parsed.

- **QCS Normalizer.** For the *m* registered services S={s_1, s_2,…, s_m}, the following matrix QCS is considered, where the *i*th row in the matrix denotes the service *i* , which consists of the four-tuple composition of its QCS properties, i.e. its Execution Duration, Price (average), Availability, and Response time, which are represented here by q_{i1} *to* q_{i4} .

$$\text{QCS:} \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ \vdots & \vdots & \vdots & \vdots \\ q_{m1} & q_{m2} & q_{m3} & q_{m4} \end{pmatrix}$$

The above matrix QCS needs to be normalized. The main reason for normalization is that different dimensions, scales and value ranges are considered for different QCS attributes, and they are not uniform. For example, the unit of measurement for response time is *millisecond* and for reliability is an integer between 1 and 24. Therefore, to develop the ranking formula, it is needed to first make a uniform quantification of for all these service qualities, independent of their measurement units. Furthermore, generating a uniform index for all service qualities also provides an equal weight for all considered criteria, as the starting point for the ranking process. Therefore, all QCS attributes are normalized into the same value range of [0,1]. In this approach, the Max-Min normalization approach is applied,

which is one of the most widely used approaches, introduced in [19], [20]. Please see Annex I for the formulas introduced to normalize the values of those QCS attributes, with positive and negative connotations.

For example, if the *response time* of three software services are 400, 450, and 510 milliseconds respectively, then these values are respectively normalized to 1, 0.54, and 0. Finally, after the normalization, the transformation matrix Q' is defined as follows:

$$Q' : \begin{pmatrix} q'_{11} & q'_{12} & q'_{13} & q'_{14} \\ \vdots & \vdots & \vdots & \vdots \\ q'_{m1} & q'_{m2} & q'_{m3} & q'_{m4} \end{pmatrix}$$

Furthermore, when the user requests a service using a multi-faceted query, it can also specify the preference requirements for the QCS metrics. These preferences are considered to be specified for $q_{i1}\ to\ q_{i4}$, and they form a weight vector, such as $W = \{w_1, w_2, w_3, w_4\}$, where $\sum_{j=1}^{4} w_j = 1$.

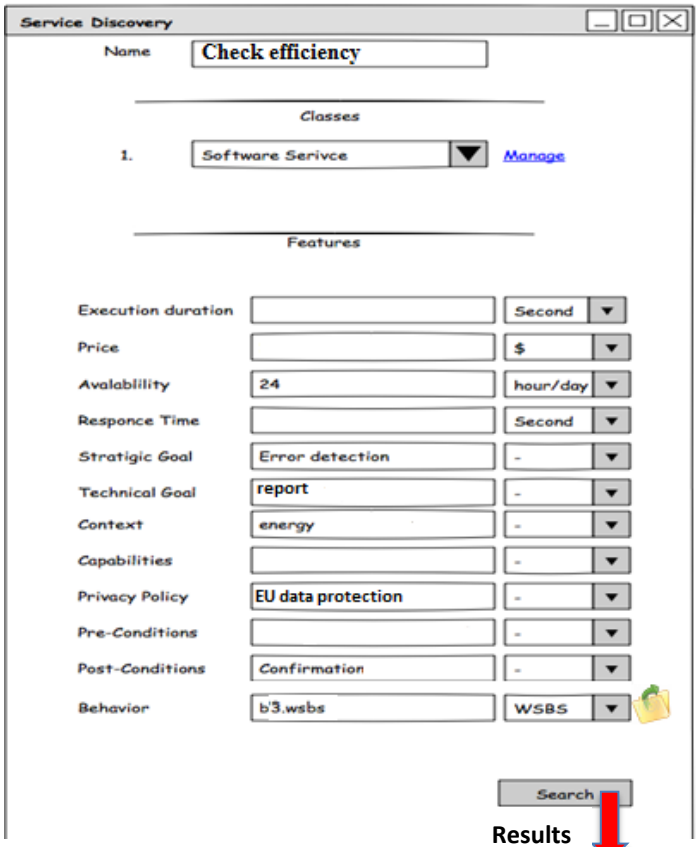
So, the comprehensive normalized QCS value for each service s_i, considering that user specifies the preference vector, is as follows:

$$Score_{QCS}(s_i) = \sum_{j=1}^{4} (w_j \times q'_{ij}),\ \ 1 \le i \le m$$

Where, *m* is the number of services.

- **Semantic Interpreter.** Besides syntactic and behavioral properties of the services, the semantics properties of the BSs (e.g. strategic goal and context) should be loaded and interpreted for textual similarities.

- **Query Processor.** At search time, a user specifies a desired service by means of a text file, and feeds it to this module. The design of an example of our multi-facet query is represented in Figure 17. The fields related to the semantics and QCS of the desired service, can be specified by simple text boxes in the suggested query form, but the user need to load a WSBS documents to identify his / her preferences for syntax and behavior aspects of the required service. The WSBS part of the query form, load a WSBS document and allows to specify all desired transitions among states, including operation names, and the names and data types of their arguments. Please note that the syntactic parameters for search (e.g. arguments' names) among service can be found in WSBS documents, therefore we do not need to ask the user about his / her syntactic preferences.



Ranking	Service ID	Service Name	Similarity Score	Address
1	241	Check & Report	0,73	www.webservicex.net/Check&Report.wsdl
2	65	Check Device	0,61	www.iplon.com/CheckDevice.wsdl
3	586	Efficiency Report	0,58	www.webservicex.net/ws/EfficiencyReport.wsdl
4	327	Monitoring Report	0,44	www.sharedservice.org/ws/a2/Monitoring.wsdl
5	811	Efficiency Test	0,42	www.prolon.com/test/Efficiency.wsdl

Figure 17. An example design for a multi-facet query form and result form, to discover a service

- **Similarity Calculator.** As Figure 16 shows, this module requires four inputs: the parsed WSDLs, WSBSs, Semantic properties and the processed query. It returns three different kinds of similarity scores, which reflect the similarities between one service and one query: i) syntactic similarity score, ii) semantic similarity score, and iii) behavioral similarity sore.

  i) Syntactic similarity score: The syntactic similarity score consists of similarities between operation names, names of input-parameters of operations, and data types of the input-parameters for one specific service and one service. We use different string similarity-metrics (also known as string distance functions) as the functions to measure the syntactic similarity between two text-strings. We have chosen three of the most widely known metrics, the *Levenshtein Distance*, the Matching Coefficient, and the *QGrams* Distance. Each of these metrics operates with two input strings, and returns a score estimating their similarity. Since each function returns a value [0..1], we average the three scores and merge them into a single [0..1] value.

  - *ii) Semantic similarity score:* The next similarity score is semantic similarity, which is calculated to find the semantic similarity between two contextual properties such as goal, using a semantic similarity function such as *wordnet* [21]. This function returns also a value in [0..1] as the estimated score.

- *iii) Behavioural similarity sore:* This score is computed to show how much two behavioral signatures of a query and one specific service are similar. The basic idea is to compute an approximate bi-simulation [22] between the two automata, respectively representing the behavior of a query and a registered service. The notion of approximate bi-simulation relation is obtained by relaxing the equality of output traces: instead of requiring them to be identical, we require that they remain "close". Metrics (represented as semirings, in our case) essentially quantify how well a system is approximated by another based on the distance between their observed behaviours. In this way, we are able to consider different transition-labels by estimating a similarity score between their operation interfaces, and different numbers of states. To model approximate bi-simulation with constraints, we exploit constraint-based graph matching techniques [23]; thus, we are able to "compress" or "dilate" one automaton structure into another.

The estimated similarity scores are subsequently used by the Constraint Assembler in Figure 16, in order to define constraints and model the CSP. In fact, the representation of the search problem in terms of constraints is completely constructed by the Constraint Assembler module, while the Similarity Calculator only provides it with similarity scores.

- **Constraint Assembler.** This module produces a model of the discovery problem, as a constraint satisfaction problem (CSP). To do so, it represents all user-preferences and similarity scores as constraints. In order to assemble these constraints, we can use JaCoP [24], which is a Java library that provides a finite-domain constraint programming paradigm. We have made ad-hoc extensions to the crisp constraints supported by JaCoP in order to equip them with weights, and we have exploited the possibility to minimise/maximise a given cost function to solve Soft CSPs (SCSPs). For instance, *SumWeight* is a JaCoP constraint that computes a weighted sum as the following pseudo-code: *w1.x1+ w2.x2+ w3.x3= sum*, where sum represents the global syntactic similarity between two operation in terms of the similarity between their operation names (x1), their argument names (x2), and their argument types (x3). These scores are provided by the *Similarity Function* module. Moreover, we can tune the weights w1, w2, and w3 to give more or less importance to the three different parameters. In this work, we use equal weights for this *SumWeight* constraint.
- **SCSP solver.** Finally, after the specification of the SCSP model in terms of variables and constraints, a search for a solution of the assembled SCSP can be started. The result can be generalized as a ranking of services in the considered database: at the top positions we find the services that are more similar to a user's request. The table in Figure 17 shows the design of search results of the example.

3.3 DISCUSSION

We have designed a tool for similarity-based discovery and ranking of most-fit business services to the required criteria specified by user. Ranking of services is performed according to the similarity score between each service and the description of a service desired by the user, i.e. through the multi-faceted query. The designed tool approaches the problem using some soft constraints, which allow to quantitatively estimate the differences between two specifications (the query specification versus service specification). Defining this problem as a SCSP (Soft Constraint Satisfaction Problem) makes the approach parametric, in respect to the chosen similarity metrics, and allows using efficient AI techniques for solving the problem.

4 SUB-PRODUCT DISCOVERY AND RANKING OF MATCHED SUGGESTIONS - PRODUCT REUSABILITY

4.1 Sub-product Search

When designing the specifications of a complex product, stakeholders (i.e. product designers) would normally not start from scratch in specifying all needed sub-products of the complex product. Different designers usually start with an existing sub-product specification and then may re-specify or customize it for the new complex product therefore, the user would be interested in looking up possibilities among the existing sub-products that are close to the new ideal sub-product that he/she wishes to specify. If a suitable sub-product matching his/her criteria is identified then it would be used as the base for his/her work on the new sub-product. Other than that, the designer of a complex product might be interested to see the specification of products existing within the VBE, what he/she can use as a sub-product, in order to build the complex product.

As a first step in this process, it is necessary to enable the user by providing search mechanisms for existing product specifications within the product specification sub-system as a part of the PSDR engine. In the upcoming sections we will describe the requirements for such search mechanisms.

Four different kinds of search mechanisms are suggested below in sections 4.1.1 to 4.1.4 addressing feature-based search, sub-product based search, class-based search and acquaintance-based search respectively.

4.1.1 Feature-based search

When dealing with a system containing multitudinous specifications and multi-disciplinary stakeholders, assuming the existence of a common terminology is unjustified. However, this lack causes a major challenge if preforming search queries based on the name of products. For example, simply considering the fact that the words “intelligent”, “smart ”, and “automated” are used in the construction industry to refer to the same concept, referring to “an object being able to vary its state or action in response to varying situations”. This situation might get even more complicated when one word might have multiple meanings in the context of a VBE, for example the word “intelligent” in the computing industry refers to a product that “incorporates a microprocessor” or a product that “has its own processing capability”.

In order to overcome this issue, we have designed, a search functionality based on sub-product features in order to search for certain products. In this approach, one can search a product specification by providing an example of what he/she expects of a product specification. This is done, by enabling the user to specify his required product through specification of some of its detail. Then the system will-lookup and discover the closest matching product specifications to the specified request.

The approach also supports the user with specifying three different types of features, namely: existence, rigid, and finally fuzzy. For existence features, the system only checks to see if the existing sub-product has the given feature-kind[1] or not. In this case the search is not sensitive to the feature itself, namely the value and unit of the feature are not checked, but it is only sensitive to the

[1] A feature-kind (e.g. weight) is a characteristic (e.g. weight) of a sub-product, which may be specified with multiple units (e.g. Kilogram, pounds, etc.) [1].

existence of a feature for a specific feature-kind. Therefore, for this approach the system looks for classes of products that have that features-kind as an obligatory feature-kind. After finding such classes the system searches for sub-product specifications that have the discovered classes, and suggests those sub-products as results. If sufficient (where the threshold will be set by the system configurator based on user evaluations) results are not found the system then looks into specifications that have the specified feature as an optional feature. This type of search is visualized in Figure 18.

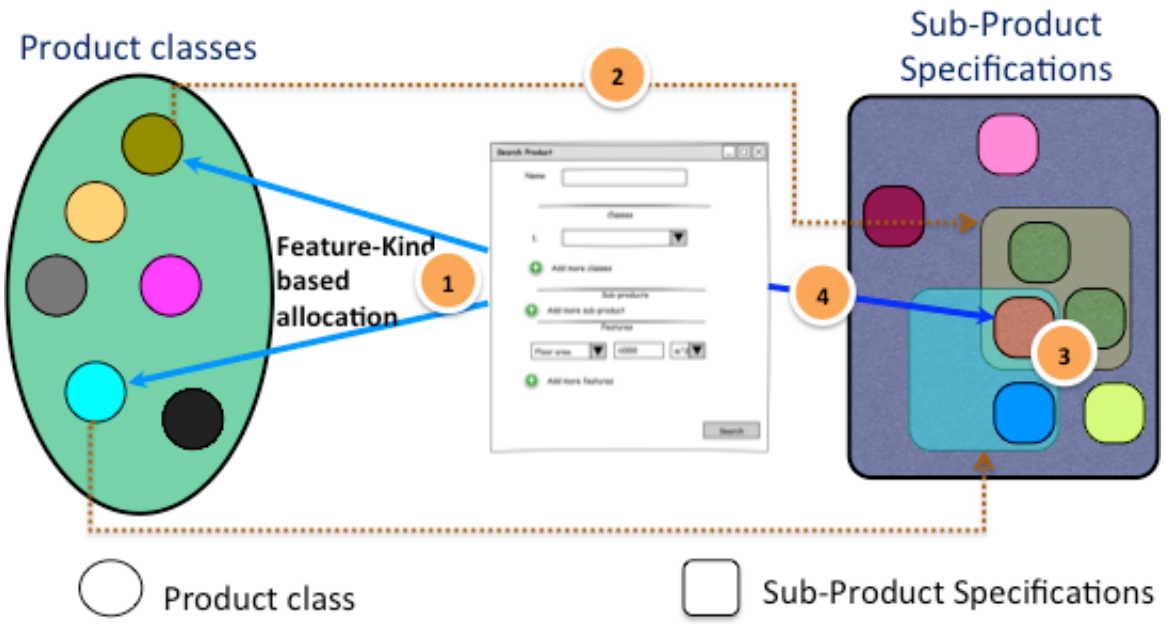


Figure 18 – Discovering similar specifications based on features

In the case that the user has selected a *rigid* feature and has also provided the value and unit of the feature, the search is then limited to sub-product specifications that both have the feature of the provided feature-kind and that the feature has the provided value and unit. It is important to point out here that only having the value (without the unit) is not sufficient for this search due to the fact that the feature might be stored with different units while representing the same amount. For example it might be the case that a feature is stored in both centimeters and meters, and 1cm or 1m are totally different values. Finally if the user has selected a *fuzzy* feature, the system will look for specifications that have the given value of the provided feature but within a range of values. The user might provide this range or the user may ask that the system to give the range while only providing an approximate value and selecting the "dynamic variation" checkbox option in the interface. The dynamic variation of values for a feature-kind is constantly calculated for each feature-kind by the system. This is described in more details in the next sub-section.

4.1.1.1 Feature-kind variation calculation

Feature-kind variation calculation is only possible when dealing with numerical feature-kinds. The variation of each numerical feature is calculated for each unit its feature-kind supports. For example the variation of length is calculated for either "centimeters" or "meters", if these two are the units defined for the feature-kind length. The based on which the variation is calculated, is what the users selects for the feature. The reason for this is twofold. First, if we calculate the variations for all the units all together we will get a very high value for the variation due to the fact that same values might show huge differences for example 1 meter will show as 100 cm. Second, a feature-kind in two

different units normally do not happen in one kind of product specification. This is due to the fact that the unit of a product is highly dependent to the so-called "Type" of product. For example, the width and height of a building is normally measured in meters while these measurements are done in centimeters for a light bulb.

The variation value is calculated based on the standard deviation of available values for a feature-kind in a specific unit as follows:

$$variation_{fk,u}(x) = \frac{\delta . SD(V_{fk,u})}{|\overline{V_{fk,u}} - x|}$$

Where fk is the feature kind, u is the unit, $V_{fk,u}$ are the possible values of a feature kind in a specific unit and δ is a constant number smaller or equal to one that is set by the system based on user feedback captured in the users profile (See section 5.2) and is one by default one.

4.1.2 Component-sub-product-based search

When looking for the most-fit existing sub-product specification, one might be interested in finding the specifications for those sub-products that contain another specific sub-product. To support this kind of requests we have designed a search mechanism where the user can search for a sub-product that contains another sub-product. This search is not limited only to the direct child of the sub-product, but can go deeper into the tree structure of the sub-product definitions. In such cases the user can actually indicate the maximum depth that the search should perform. For example the user might be interested in sub-products that have a lamp sub-product specification in a maximum depth of 3 in their sub-product tree. This search is in fact beneficial when the user is interested in the specifications of the surrounding elements of a sub-product. As such in the case of the lamp you can actually find out the specifications of where this specific lamp is used by looking up sub-products that had the sub-product lamp as a constituting sub-product. With this search the user can find answers to his questions such as "Is this sub-product more suited for outdoors or indoors?" This is done by searching for sub-product specification that have the sub-product as their constituting sub-product and then looking into if that sub-product is outdoor or indoor (e.g. is the parent sub-product a room or not).

4.1.3 Class-based search

Although in most cases the user might know the characteristics of a sub-product for which he/she is looking, it might also be the case that the user is only able to guess the class of the sub-products. In this case it is possible to indicate the specific classes that a product belongs to within the query. Other than having user defined classes we also introduce and, use the PRODCOM [25] convention for classifications, that are pre-defined in the system that makes the assigning of classes as well as the search process for sub-products easier. This is due to the fact that using a standard taxonomy as the base for product classification, converging the terminology and taxonomy of the different disciplines into one common multi-disciplinary terminology and taxonomy. The reason we choose PRODCOM [25] is twofold. First, it is an agreed standard within the European community as the name "PRODucts of the European COMmunity" also suggests. PRODCOM [25] is actually a yearly extension to the well-known CPA 2008 [26] (Statistical Classification of ***Products by Activity*** in the European Economic Community, 2008 version) classification. Second, not only is PRODCOM [25] extendable but due to the fact that it is an extension to CPA [26] it has also an agreement with another very important coding system, which is the NACE Rev. 2 [17]. The NACE Rev. 2 coding system as its

description, "Statistical Classification of ***Economic Activities*** in the European Community" suggests, indicates the economic activity that is performed in order to build a product. This comes handy later on, for recommending services that enhance sub-products and provides valuable input when a specification is being realized by the VO formation sub-system and organizations need to be selected for VO by the VO planner of the complex product to perform the VO-related tasks.

4.1.4 Acquaintance-based search

The above mentioned search methods all focus on a sub-product being stand-alone. In this search mechanism the user can query for a sub-product specification that are previously specified within the hierarchal structure of sub-products previously defined in a complex product. In this case you can for example look-up specifications of lamps that have been used in a power plant specification before or you can look up light sensors specifications that have been used in the same complex product specification as a specific type of solar panel specification exists.

4.2 Suggestions in the process of sub-product specification

Suggestion, also known as recommendation, is an effort to filter information and personalize the support provided to the user [27]. A recommender system for products and services is defined as the system, which recommends an appropriate sub-product or service after considering the set of user's preferences and desires [28]. These systems bring several advantages that include the increase of cross selling and user satisfaction. Other than this, such systems fulfill user needs by presenting side products of possible interest to them [29].

In the case of the complex product specification sub-system we provide suggestions in order to enable reusability in deferent different points. These suggestions include providing prospective features, sub-products and classes for a composite sub-product. The suggestions also include providing alternative sub-product specifications from existing specifications. At the end, the product specification sub-system also provides a set of advanced suggestion functionalities for housekeeping of the system (e.g. garbage collection). In this section we will provide details of how and where each suggestion is provided and how suggestions are ranked to be presentation to the user. The upcoming sections (4.2.1 to 4.2.6) will describe our provided suggestions, while only the 4.2.1 repeats all needed calculations, you can find an extended version of the complete section 4.2 in Annex II.

4.2.1 Suggestion of constituting sub-products for a composite sub-product

One of the dimensions of reusability is providing the possibility to extend an existing product specification. But before preforming the duplications process one should find the suitable specification to duplicate. Although the search mechanisms provide the possibility to search within the existing specifications, this may not lead the user to his/her desired specification to start from. To provide further options after the user has selected a specification and is viewing the specification to verify if this fits his/her needs. The system provides suggestions of similar specifications in the view screen that the user can choose to view. This way the user gets closer to his/her desired specification more quickly. The suggestions presented here are sorted based on the ranking function, $Rank_{p,v}\ (a, b)$. This ranking function is based on the similarity of two sub-products ($sim_p(a, b)$).

Similarity of two sub-products: Two sub-products are similar when they have the following conditions: 1) Their names are similar, 2) they are from similar classes, 3) they have similar features and, 4) they are constituted of similar sub-products. For each of the mentioned cases above we calculate a

similarity function and finally merge the functions by preforming a weighted average between the functions to form the final similarity function.

For similarity in name we adopt a wordnet based method that is an extended version of the WUP semantic similarity approach [30] and has been proposed by Resnik[31].

$$WS_{pn}(x) = \{\, w\text{: } w \; is \; a \; word \; used \; in \; name(x) \;\; where \; x \; is \; a \; product\}$$

$$Q_{pn}(a,b) = \{q\text{: } q = WUP(s_1, s_2); and \; s_1 \in WS_{pn}(a); and \; s_2 \in WS_{pn}(b);\}$$

$$sim_{pn}(a,b) = Max\big(Q_{pn}(a,b)\big) \; where \; a, b \; are \; products$$

For similarity using class similarity we can use:

$$sim_{pc}(a,b) = \frac{|classes(a) \cap classes(b)|}{|classes(a) \cup classes(b)|} \; where \; a \; and \; b \; are \; products$$

For similarity using feature similarity we can use:

$$sim_{pf}(a,b) = \frac{|features(a) \cap features(b)|}{|features(a) \cup features(b)|} \; where \; a \; and \; b \; are \; products$$

And finally for similarity using sub-product similarity we can use:

$$sim_{ps}(a,b) = \frac{|subProducts(a) \cap subProducts(b)|}{|subProducts(a) \cup subProducts(b)|} \; where \; a \; and \; b \; are \; products$$

Now based on the similarities above we can calculate the similarity between two sub-products as:

$$sim_{p}(a,b) = \frac{K_1.sim_{pn}(a,b) + \; K_2 sim_{pc}(a,b) + K_3 sim_{pf}(a,b) + K_4 sim_{ps}(a,b)}{K_1 + K_2 + K_3 + K_4}$$

$$where \; K_1, K_2, K_3 \; and \; K_4 \;\; are \; constants \; with \; the \; default \; value \; of \; one$$

The constants can be tuned based on the preference a user has on one type of similarity to another. This can be indirectly inferred from the users interaction with the system.

This similarity function can actually be used as the ranking function for suggestions when viewing a sub-product specification:

$$Rank_{p,v}\,(a,b) = sim_p(a,b)$$

Suggestions while viewing a sub-product specification are not the only suggestion that can be provided to the user to enable reusability of sub-products specifications. It might be the case that an existing sub-product specification is not used as a base of an extended specification but rather it is used as a sub-product constituting a new composed specification. In this case the system should provide suggestions of suitable sub-products that can be part of the composed sub-product under specification. In such cases the suggestions are provided in the registration step of product specification. The ranking of this kind of suggestion ($Rank_{p,sp}(a)$) is based on the likelihood of a sub-product being in a setting of a composed sub-product. We call these the Likelihood of acquaintance and they are as follows:

Likelihood of acquaintance for sub-products: Two sub-products are acquainted if they are sub-products of a common composed/complex product. The likelihood of their acquaintance depends on two parameters: 1) the amount of times they have constituted a common sub-product, and 2) the average distance between the sub-products in the complex product trees. Please note that each sub-product might participate in multiple complex product trees or in other words a sub-product might be a constituting sub-product of multiple complex products.

In order to calculate these likelihoods we should know what the common ancestors ($CA_{p,t}(a,b)$) of two sub-products are. The common ancestors of two sub-products in a complex product tree are the ancestors that if removed from the complex product tree will remove the acquaintance of the two sub-products in that complex product tree. Please not that if a sub-product has been used more than once as a constituting sub-product of a complex product then that sub-product will have more than one common ancestor with other sub-products:

$$A_{p,t}(a) = \{x: x \in parents(a)\ ; or\ x \in A_{p,t}(parent(t))\ where\ t \in parents(a)\}$$

$$CA_{p,t}(a,b) = A_{p,t}(a) \cap A_{p,t}(b)$$

So the first parameter can me measure by:

$$LA_{pi}(a,b) = \frac{|CA_{p,t}(a,b)|}{|products|}$$

For the second parameter we can do a calculations as follows:

$$depth_p(a,b,c) = \begin{cases} \{(c+1)\} \cup depth(childeren(b), a, c+1)\ if\ a \in children(b) \\ depth_p(children(b), a, c+1)\ if\ a \notin children(b)\ and\ children(b) \neq \emptyset \\ \emptyset\ if\ children(b) = \emptyset \end{cases}$$

$$where\ a\ is\ a\ subProduct\ of\ b; and\ c\ is\ zero\ at\ the\ start$$

$$AD_p(a,b) = average(\{x = depth(a,x,0) + depth(b,x,0): where\ x \in CA_{p,t}(a,b)\}$$

$$LA_{pd}(a,b) = \begin{cases} 1 \; if \; AD_p(a,b) \leq 2 \\ \dfrac{2}{AD_p(a,b)} \; if \; AD_p(a,b) > 2 \end{cases}$$

Based on the two calculated parameters the likelihood of acquaintance of two products can be calculated as:

$$LA_p(a,b) = \frac{K_1.LA_{pi}(a,b) + K_2 LA_{pd}(a,b)}{K_1 + K_2}$$

$$where \; K_1 and \; K_2 \; are \; constants \; with \; the \; default \; value \; of \; one$$

The constants can be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred from the users interaction with the system.

Likelihood of acquaintance of a sub-product with a feature: A sub-product and a feature are acquainted if the feature is a feature of a sub-product that constitutes the sub-product. We can calculate the likelihood using:

$$LA_{fp,p}(x,y) = \frac{|\{p: x \in \; features(p) \; and \; y \in subProducts(p) \; where \; p \in products \}|}{|products|}$$

Likelihood of acquaintance of a sub-product with a feature-kind: A sub-product and a feature-kind are acquainted if a feature of the kind of that feature-kind is a feature of a sub-product and that sub-product is composed of the sub-product. We can calculate the likelihood using:

$$LA_{fkp,p}(x,y) = \frac{|\{p: x \in \; PFK(p) \; and \; y \in subProducts(p) \; where \; p \in products \}|}{|products|}$$

Likelihood of acquaintance of a sub-product with a class: A sub-product and a class are acquainted if the class is a class of a sub-product and that sub-product is composed of the sub-product. We can calculate the likelihood using:

$$LA_{p,cp}(x,y) = \frac{|\{p: x \in \; subProducts(p) \; and \; y \in classes(p) \; where \; p \in products \}|}{|products|}$$

Now based on the likelihoods calculated above we can rank suggestions of sub-products that constitute another sub-product as follows, please note that a is a candidate sub-product:

$$A_1(a) = average(LA_p(a,b_1), \ldots, LA_p(a,b_n))$$

$$where\ b_1, \dots, b_n\ are\ candidate\ subProducts\ for\ the\ new\ product,$$

$$A_2(a) = average(\ LA_{fp,p}(c_1, a), \dots, LA_{fp,p}(c_n, a))$$

$$where\ c_1, \dots, c_n\ are\ candidate\ features\ for\ the\ new\ product$$

$$A_3(a) = average(\ LA_{fkp,p}(d_1, a), \dots, LA_{fkp,p}(d_n, a))$$

$$where\ d_1, \dots, d_n\ are\ candidate\ featureKinds\ for\ the\ new\ product$$

$$A_4(a) = average(LA_{p,cp}(a, e_1), \dots, LA_{p,cp}(a, e_n))$$

$$where\ e_1, \dots, e_n\ are\ candidate\ classes\ for\ the\ new\ product$$

$$Rank_{p,sp}(a) = \frac{K_1A_1(a) + \ K_2A_2(a) + K_3A_3(a) + K_4A_4(a)}{K_1 + K_2 + K_3 + K_4}$$

$$where\ k_1, \dots, k_n\ are\ constant\ and\ are\ one\ by\ default$$

The constants can be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred from the users' interaction with the system.

4.2.2 Feature suggestion for a sub-product

Not only should we have reusability for product specifications but also for the building components of the specification. Features are the building blocks of specifications and reusing them enables both reduction in space and also a unified definition for specifications. When specifying a sub-product, each user has different views on what the features of a sub-product are. By providing suggestions for features in the process of sub-product suggestion we speed-up the specification process and guide the user on how he should specify a sub-product.

The ranking for a feature in such suggestion is based on the likelihood of a feature being in a setting of a sub-product. Other that the likelihood of acquaintance of a sub-product with a feature in the case of composed products.

4.2.3 Features-kind suggestion for a sub-product

While suggestions of features are suitable to quickly converge the user to a specific specification it might be the case that suggestions provided with that level of details are not useful to the user as he or she is specifying an innovative specification that has not been seen in the system before. As such just recommending feature-kinds that should be covered for the sub-product might come very handy. We can rank existing feature-kinds in the system in order to suggest to the user in the process of specifying a sub-product. This ranking is based on the likelihood of the feature-kind showing up in different settings.

4.2.4 Class suggestion for a sub-product

In the process of guiding the user to specify sub-products, an important part is to suggest the user with existing classes that the sub-product specification could be classified under. This suggestion is an important step due to the fact that users normally are not fully aware of possible options for classes that the product specification might fall under. The ranking for this suggestion is based on the likelihoods for a class to be present in a configuration.

4.2.5 Obligatory features-kind suggestion for a class

In order to enhance the user's experience and to guide the user in the specification process, when adding a new class obligatory feature-kinds are recommended to the user.

4.2.6 Garbage collection and duplicate prevention

When adding new entities, it is important to prevent the user from creating duplicate entities. Capturing duplicates is a difficult task when there are ambiguities. In order to prevent ambiguities for example in names we calculate the similarity of a new entity with existing entitles and if the similarity of the entity to an existing entity is more than a threshold, the system will present a warning to the user and will suggest the existing entity instead of adding a new one.

5 SUPPORTING SERVICE-ENHANCED PRODUCT RECOMMENDATION

As mentioned before, enhancing a product with services plays an increasingly important role in today's knowledge-based economy especially in the field of intelligent buildings and solar power plants. These services that provide enhanced functionality for the products differentiate them from similar existing products in the market. Such enhancement not only provides a higher level of differentiation from the competing similar products in the market, but it also increases the value of the products that will be later realized. An example of such enhancing service in GloNet is adding software services to monitor solar panels in solar power plant (e.g. a notification based monitoring service that generates an alert when production goes below the threshold, and a report generation monitoring service that generates a weekly report based on the production status of the solar panel). As another example of such enhancing services we can consider adding the manual tasks of yearly maintenance of office lamps, being specified for an intelligent building. These lamps can therefore, not only have enhanced software services for setting their brightness, but also have this manual service to support their maintenance. Figure 19 shows an image of such lamps.

Figure 19 – Lamps in an intelligent building

In order to benefit from enhancements of products with services, a designer of the complex product would like to find any and all related previously specified services that could be potentially added/attached to the sub-product that is currently under specification for the complex product.

Although providing such users with the list of existing services could be a solution to encourage users to enhance their sup-products with services, providing such a list may cause the common known problem of information overload. In such a case users would not be able to easily find what suites them in a sufficiently short amount of time and therefore, they may feel lost or lose trust during the process of searching for an enhancing service. This is mostly due to the fact that a long list of existing services, with similar names, makes it difficult for the user to judge how reliable or relevant each existing service is for their sub-product.

Applying Recommender systems is a known and effective solution for such cases. A **Recommender System** is "a system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way, to interesting or useful objects in a large space of possible options" [32]. These systems are commonly used in commercial software systems to help them with making their decisions. The use of such systems will increase the interest of the user to have

productive interactions with the system, and will increase the users' satisfaction through "creating a value-added relationship between the system and the user" [33].

This can be realized by providing recommendations of possible enhancing services to the user when he or she is specifying a sub-product. Recommender systems such as the one designed within the PSDR engine mainly deal with two types of information namely, relatedness/acquaintance of the entities they would like to provide recommendation on and the user behavior. In the upcoming sections we will look into how we consider a product and services to be related. Then we will look into how we profile the users' interactions with the system in order to improve future recommendations.

5.1 Standard-based recommendation

In order to recommend services that enhance a product we can use the classes of the products and services in order to build a link. As we have mentioned before, sub-products can be classified using the PRODCOM [25] coding system in the system. Other than this, services can be classified using the NACE [17] coding system provided by the system. Due to the nature of the PRODCOM coding system [25] as shown in Figure 20, the PRODCOM coding system [25] embeds within itself the activity associated to the product. If a product has at least a class that is a PRODCOM class (classes with the PRODCOM code) we can use this class and it's coding in order to find suitable product by looking into the NACE [17] part of the PRODCOM [25] coding and looking for services with this NACE [17] coding. This mechanism enables suggesting a set of enhancing services for a product. We follow the approach suggested by Resnik [31] to attach a rank for the similarity of the product classes NACE code [17] (that is part of the PRODCOM code) with the NACE code [17] of services. If a sub-product or service has more than one class then the maximum of the similarity between the classes of the sub-product and service is considered as the similarity of the service and sub-product. Please note that due to the nature of the NACE coding system [17], when a sub-product and service have a similar coding it is not that they have similar specifications. This similarity shows that the sub-product and service belong to the same sector (e.g. the NACE code F42.1.3 refers to Construction of bridges and tunnels that is more or less a sector of activity than a very specific activity).

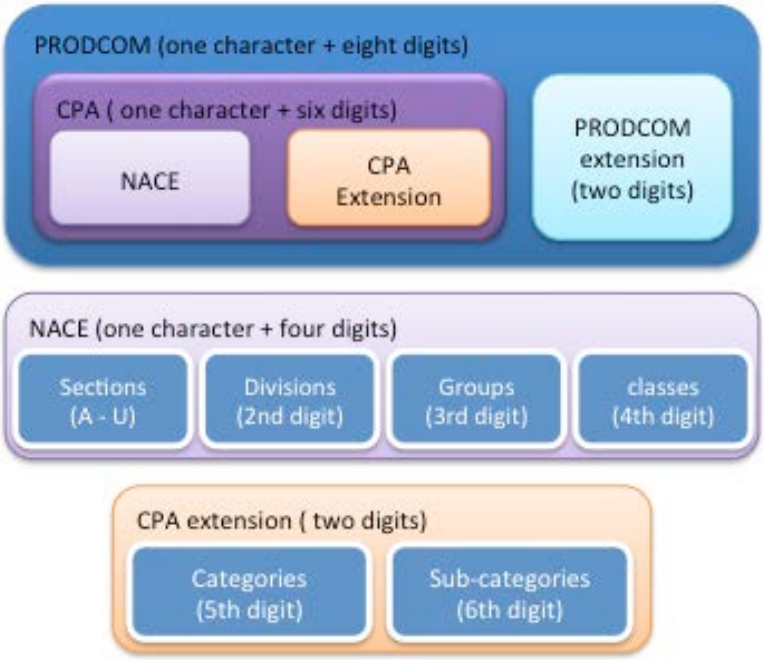


Figure 20 – The structure of EU standard classification systems

5.2 Specification-based recommendation

Other than following a standard-based approach for recommending services enhancing a sub-product, we can alternatively use as an approach similar to the one suggested in section 4.2.1, where we described how we suggest constituting smaller sub-products for a composite sub-product. For specification-based recommendation of services enhancing the sub-product, we assume that an enhancing service is a constituting sub-product. Then using the same similarity functions mentioned before in that section we can find similar sub-products to the one currently being specified by the user. Furthermore, checking on the services that have been previously specified for enhancing those similar sub-products, the sub-system suggest the same associated services for the current sub-product being specified. Nevertheless, clearly the final choice of the accepting our recommended services lie with the user of the system.

5.3 User/Usage profile-based recommendation

When recommending services enhancing sub-products using the method mentioned above, due to the fact that NACE codes [17] have a broad definition, there might be multiple services suitable for recommending as enhancers of a sub-product. In order to rank these services more properly and more tailored to the user we use the users' interactions with the system to capture indirect/passive feedback form him or her. This feedback is then used to identify the preference of the user for a specific service and to identify how the user looks as relationships between classes of products and services. Capturing the users' interactions is realised using the users profile.

5.3.1 User's usage profile

In order to provide the users with specific recommendations we build a usage profile using all the interactions the user preforms with the system and visa versa. This is used to later analyze his/her preferences. This profile consists of all actions the user has performed on entitles. The table below shows the fields of the stored data.

Field	Discription
UserID	The ID of the use preforming the action
EntityID	The ID of the entity that is evolved in the action for example when a user views and entity the ID is recorder here
EntityType	The type of the entity for example product, class, featureKind, feature, etc.
Action	The action that is performed on the entity for example view, duplicate, addAsSubProduct, etc.
TargetEntityID	The ID of the entity that the target of the action for example when a user adds a subProduct to a product the id of the product is recorder here
TargetEntityType	The type of the TargetEntity
DirectoryID	The ID of the directory where this action is being performed, if available
DGID	The ID of design group where this action is preformed, if available
TimeStamp	The time the action was preformed

It is important to know that the actions that the system preforms for a user is also recorded in this profile. For example recommendations that are provided are also recorded in the profile.

Using this profile, through some statistical analysis, we can identify what recommendations are accepted or rejected by the user or what specifications are more frequently used (added as constituting sub-product, added as enhancing service etc.) by the user [34]. In the most simple case this usage profile will be used for recommending the most frequently constituting sub-product/enhancing service selected by the user. Other than this, the profile can also be used to figure out the preferred constant values that have been introduced previously for sub-product recommendation for the user. This is done by counting the number of chosen entities that have a higher value in one of the likelihoods or similarities for that user [34].

5.3.2 Design Group's usage profile

In order to provide the users with design group specific recommendation we should have a profile to later analyze the design group's preferences. This profile is actually a subset of the combination of all user profiles. The table below shows fields that are available for this profile:

Field	Discription
DGID	The ID of the design group that this action is preformed within, if available
EntityID	The ID of the entity that is evolved in the action for example when a user views and entity the ID is recorder here
EntityType	The type of the entity for example product, class, featureKind, feature, etc.
Action	The action that is performed on the entity for example view, duplicate, addAsSubProduct, etc.
TargetEntityID	The ID of the entity that the target of the action for example when a user adds a subProduct to a product the id of the product is recorder here
TargetEntityType	The type of the TargetEntity
TimeStamp	The time the action was preformed

5.3.3 Directory's usage profile

In order to provide the users with directory specific recommendation we should have a profile to later analyze directory's preferences. This profile is actually a subset of each users' profile. This is due to the fact that the directory is a subset of the each users' design space. The table below shows fields that are available for this profile:

Field	Discription
UserID	The ID of the use preforming the action
DirectoryID	The ID of the project that this action is being performed within, if available
EntityID	The ID of the entity that is evolved in the action for example when a user views and entity the ID is recorder here
EntityType	The type of the entity for example product, class, featureKind, feature, etc.
Action	The action that is performed on the entity for example view, duplicate, addAsSubProduct, etc.
TargetEntityID	The ID of the entity that the target of the action for example when a user adds a subProduct to a product the id of the product is recorder here
TargetEntityType	The type of the TargetEntity
TimeStamp	The time the action was preformed

6 CONCLUDING REMARKS

In order to enhance the user's experience in specifying a service-enhanced product in different domains, such as those introduced in the GloNet project (solar power plants and intelligent buildings), we have designed a set of functionalities facilitating the complex product specification for different types of users in this environment. These functionalities include specific support for service specification & registration and Product/Service Discovery & Recommendation. As such, on one hand the reusability of the specified services is facilitated and on the other hand the process of service specification as well as enhancing complex products with their relevant services are supported.

We have devoted most of this deliverable to the parameters and mechanisms that are needed for product and service discovery and recommendation. Furthermore, we have provided a description of the users' profile, which will be also used for the purpose of customized recommendation.

7 REFRENCES

1. Afsarmanesh, H., & Shafahi, M. (2013). Specification and Configuration of Customized Complex Products. In Collaborative Systems for Reindustrialization (pp. 81-90). Springer Berlin Heidelberg. (Partially presented in D4.1)

2. Shafahi, M. (2013), Afsarmanesh, H., & Sargolzaei, M. A Coopetition Space for Complex Product Specification. In Collaborative Systems for Smart Networked Environments (15th IFIP International Working Conference on Virtual Enterprises). (Partially presented in D4.2)

3. Hertog, P. D. (2000). Knowledge-intensive business services as co-producers of innovation. International Journal of Innovation Management, 4(04), 491-528.

4. Bitner, M. J., & Brown, S. W. (2006). The evolution and discovery of services science in business schools. Communications of the ACM, 49(7), 73-78.

5. Afsarmanesh, H., Sargolzaei, M., & Shadi, M. (2014). Semi-automated Software Service Integration in Virtual Organizations. International Journal of Enterprise Information Systems "Service-based Interoperability and Collaboration for Enterprise Networks", (in print).

6. Hertog, P. D. (2000). Knowledge-intensive business services as co-producers of innovation. International Journal of Innovation Management, 4(04), 491-528

7. Baier, C., Sirjani, M., Arbab, F., and Rutten, J., 2006."Modeling Component Connectors in Reo by Constraint Automata". Science of Computer Programming, Elsevier 61(2): 75-113.

8. Camarinha-Matos, L. M., Macedo, P., Afsarmanesh, H., & Sargolzaei, M. (2013). D2.4- Mechanisms for defining composed services to support collaboration.

9. Hill, T. P. (1977). On goods and services. The Review of Income and Wealth 23(4).

10. Camarinha-Matos, L. M., Afsarmanesh, H., Oliveira, A. I., & Ferrada, F. (2013). Collaborative Business Services Provision. In Proceedings of ICEIS'13–15th International Conference on Enterprise Information Systems (Vol. 2, pp. 382-392).

11. Web Service Description Language (2001). URL http://www.w3.org/TR/wsdl

12. Simple Object Access Protocol (2000). URL http://www.w3.org/2000/xp/Group/

13. Afsarmanesh, H., Sargolzaei, M., & Shadi, M. (2012). A framework for automated service composition in collaborative networks. In Collaborative Networks in the Internet of Services (pp. 63-73). Springer Berlin Heidelberg.

14. Jongmans, S.S., Santini, F., Sargolzaei, M., Arbab, F., and Afsarmanesh, H., 2012. "Automatic Code Generation for the Orchestration of Web Services with Reo". European Conference on Service-Oriented and Cloud Computing, 19-21 September 2012, Bertinoro, Italy.

15. Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., Youssef, A.: Web services on demand: Wsla-driven automated management. IBM systems journal 43(1), 136–158 (2004).

16. Shadi, M. and Afsarmanesh, H., 2013a. "Agent Behavior Monitoring in Virtual Organizations". In Proceedings of the 22th IEEE International Workshops on Enabling Technologies (WETICE 2013), 17-20 June 2013, Hammamet, Tunisia.

17. NACE: http://epp.eurostat.ec.europa.eu/portal/page/portal/nace_rev2/introduction

18. Sargolzaei, M., Santini, F., Arbab, F., & Afsarmanesh, H. (2013). A tool for behaviour-based discovery of approximately matching web services. In Software Engineering and Formal Methods (SEFM 2013), LNCS 8137 (pp. 152-166). Springer Berlin Heidelberg.

19. Li, Y., Zhou, M., Li, R., Cao, D., Mei, H., 2008."Service Selection Approach Considering the Trustworthiness of QoS Data", Journal of software, Vol.19,No.10, pp.2620-2627.

20. Yanwei, Zh., Hong, H., Haojiang, D., Lei, L., 2010. "A dynamic web services selection based on decomposition of global QoS constraints", IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT), Nov 2010, pp.77-80.

21. George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

22. Girard, A., Pappas, G.J.: Approximation metrics for discrete and continuous systems. IEEE Trans. Automat. Contr. 52(5), 782–798 (2007)

23.. le Cl´ement de Saint-Marcq, V., Deville, Y., Solnon, C.: Constraint-based graph matching. In: Gent, I.P. (ed.) CP. Lecture Notes in Computer Science, vol. 5732, pp. 274–288. Springer (2009)

24. Java Constraint Programming solver (JaCoP): http://www.jacop.eu

25. PRODCOM: http://epp.eurostat.ec.europa.eu/portal/page/portal/prodcom/introduction

26. CPA: http://epp.eurostat.ec.europa.eu/portal/page/portal/cpa_2008/introduction

27. Lihua, W., Lu, L., Jing, L., & Zongyong, L. (2005). Modeling user multiple interests by an improved GCS approach. Expert Systems with Applications, 29(4), 757-767.

28. Choi, S. H., Kang, S., & Jeon, Y. J. (2006). Personalized recommendation system based on product specification values. Expert Systems with Applications, 31(3), 607-616.

29. Senecal, S., & Nantel, J. (2004). The influence of online product recommendations on consumers' online choices. Journal of retailing, 80(2), 159-169.

30. Wu, Z., & Palmer, M. (1994, June). Verbs semantics and lexical selection. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics (pp. 133-138). Association for Computational Linguistics.

31. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370..

32. Burke, R., Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 2002. Volume 12(Number 4): p. 331-370.

33. Schafer, J. B., Konstan, J., & Riedl, J. (1999, November). Recommender systems in e-commerce. In Proceedings of the 1st ACM conference on Electronic commerce (pp. 158-166). ACM.

34. Agosti, M., Crivellari, F., & Di Nunzio, G. M. (2012). Web log analysis: a review of a decade of studies about information acquisition, inspection and interpretation of user interaction. Data Mining and Knowledge Discovery, 24(3), 663-696.

ANNEX I

Below, in formula (1), the q' normalizes the values for those QCS attributes that have positive connotations, i.e. the Availability whose value will be scaled as follows:

$$q'_{kj} = \begin{cases} \dfrac{q_{kj} - {}_{i=1}^{m}min(q_{ij})}{{}_{i=1}^{m}max(q_{ij}) - {}_{i=1}^{m}min(q_{ij})} & if\ {}_{i=1}^{m}max(q_{ij}) \neq {}_{i=1}^{m}min(q_{ij}) \\ 1 & if\ {}_{i=1}^{m}max(q_{ij}) = {}_{i=1}^{m}min(q_{ij}) \end{cases} \quad (1)$$

Where *m* is the number of services, and $max(q_{ij})$ and $min(q_{ij})$ respectively show the maximum and minimum values in the entire column j in matrix QCS. For example, if three services are registered, and the value for the *Availability* attribute of these three services are 10, 13, and 17 percent respectively, then these values are respectively normalized to 0, 0.43, and 1.

To normalize the attributes with negative connotation, i.e. *Price, Execution Duration,* and *Response time,* the formula (2) below is used:

$$q'_{kj} = \begin{cases} \dfrac{{}_{i=1}^{m}max(q_{ij}) - q_{kj}}{{}_{i=1}^{m}max(q_{ij}) - {}_{i=1}^{m}min(q_{ij})} & if\ {}_{i=1}^{m}max(q_{ij}) \neq {}_{i=1}^{m}min(q_{ij}) \\ 1 & if\ {}_{i=1}^{m}max(q_{ij}) = {}_{i=1}^{m}min(q_{ij}) \end{cases} \quad (2)$$

ANNEX II - Suggestions in the process of sub-product specification

Suggestion, also known as recommendation, is an effort to filter information and personalize the support provided to the user [27]. A recommender system for products and services is defined as the system, which recommends an appropriate sub-product or service after considering the set of user's preferences and desires [28]. These systems bring several advantages that include the increase of cross selling and user satisfaction. Other than this, such systems fulfill user needs by presenting side products of possible interest to them [279].

In the case of the complex product specification sub-system we provide suggestions in order to enable reusability in deferent different points. These suggestions include providing prospective features, sub-products and classes for a composite sub-product. The suggestions also include providing alternative sub-product specifications from existing specifications. At the end, the product specification sub-system also provides a set of advanced suggestion functionalities for housekeeping of the system (e.g. garbage collection). In this section we will provide details of how and where each suggestion is provided and how suggestions are ranked to be presentation to the user. The upcoming sections will describe the provided suggestions, you can find an extended version of this chapter in Annex B.

1. Suggestion of constituting sub-products for a composite sub-product

One of the dimensions of reusability is providing the possibility to extend an existing product specification. But before preforming the duplications process one should find the suitable specification to duplicate. Although the search mechanisms provide the possibility to search within the existing specifications, this may not lead the user to his/her desired specification to start from. To provide further options after the user has selected a specification and is viewing the specification to verify if this fits his/her needs. The system provides suggestions of similar specifications in the view screen that the user can choose to view. This way the user gets closer to his/her desired specification more quickly. The suggestions presented here are sorted based on the ranking function, $Rank_{p,v}\ (a, b)$. This ranking function is based on the similarity of two sub-products ($sim_p(a, b)$).

Similarity of two sub-products: Two sub-products are similar when they have the following conditions: 1) Their names are similar, 2) they are from similar classes, 3) they have similar features and, 4) they are constituted of similar sub-products. For each of the mentioned cases above we calculate a similarity function and finally merge the functions by preforming a weighted average between the functions to form the final similarity function.

For similarity in name we adopt a wordnet based method that is an extended version of the WUP semantic similarity approach [30] and has been proposed by Resnik[31].

$$WS_{pn}(x) = \{\, w\text{: } w \; is \; a \; word \; used \; in \; name(x) \;\; where \; x \; is \; a \; product\}$$

$$Q_{pn}(a, b) = \{q\text{: } q = WUP(s_1, s_2); and \; s_1 \in WS_{pn}(a); and \; s_2 \in WS_{pn}(b);\}$$

$$sim_{pn}(a, b) = Max\big(Q_{pn}(a, b)\big) \; where \; a, b \; are \; products$$

For similarity using class similarity we can use:

$$sim_{pc}(a,b) = \frac{|classes(a) \cap classes(b)|}{|classes(a) \cup classes(b)|} \; where\ a\ and\ b\ are\ products$$

For similarity using feature similarity we can use:

$$sim_{pf}(a,b) = \frac{|features(a) \cap features(b)|}{|features(a) \cup features(b)|} \; where\ a\ and\ b\ are\ products$$

And finally for similarity using sub-product similarity we can use:

$$sim_{ps}(a,b) = \frac{|subProducts(a) \cap subProducts(b)|}{|subProducts(a) \cup subProducts(b)|} \; where\ a\ and\ b\ are\ products$$

Now based on the similarities above we can calculate the similarity between two sub-products as:

$$sim_{p}(a,b) = \frac{K_1.sim_{pn}(a,b) + K_2 sim_{pc}(a,b) + K_3 sim_{pf}(a,b) + K_4 sim_{ps}(a,b)}{K_1 + K_2 + K_3 + K_4}$$

$$where\ K_1, K_2, K_3\ and\ K_4\ \ are\ constants\ with\ the\ default\ value\ of\ one$$

The constants can be tuned based on the preference a user has on one type of similarity to another. This can be indirectly inferred form the users interaction with the system.

This similarity function can actually be used as the ranking function for suggestions when viewing a sub-product specification:

$$Rank_{p,v}\ (a,b) = sim_{p}(a,b)$$

Suggestions while viewing a sub-product specification are not the only suggestion that can be provided to the user to enable reusability of sub-products specifications. It might be the case that an existing sub-product specification is not used as a base of an extended specification but rather it is used as a sub-product constituting a new composed specification. In this case the system should provide suggestions of suitable sub-products that can be part of the composed sub-product under specification. In such cases the suggestions are provided in the registration step of product specification. The ranking of this kind of suggestion ($Rank_{p,sp}(a)$) is based on the likelihood of a sub-product being in a setting of a composed sub-product. We call these the Likelihood of acquaintance and they are as follows:

Likelihood of acquaintance for sub-products: Two sub-products are acquainted if they are sub-products of a common composed/complex product. The likelihood of their acquaintance depends on two parameters: 1) the amount of times they have constituted a common sub-product, and 2) the average distance between the sub-products in the complex product trees. Please note that each sub-

product might participate in multiple complex product trees or in other words a sub-product might be a constituting sub-product of multiple complex products.

In order to calculate these likelihoods we should know what the common ancestors ($CA_{p,t}(a,b)$) of two sub-products are. The common ancestors of two sub-products in a complex product tree are the ancestors that if removed form the complex product tree will remove the acquaintance of the two sub-products in that complex product tree. Please not that if a sub-product has been used more that once as a constituting sub-product of a complex product then that sub-product will have more that one common ancestor with other sub-products:

$$A_{p,t}(a) = \{x: x \in parents(a)\ ; or\ x \in A_{p,t}(parent(t))\ where\ t \in parents(a)\}$$

$$CA_{p,t}(a,b) = A_{p,t}(a) \cap A_{p,t}(b)$$

So the first parameter can me measure by:

$$LA_{pi}(a,b) = \frac{|CA_{p,t}(a,b)|}{|products|}$$

For the second parameter we can do a calculations as follows:

$$depth_p(a,b,c) = \begin{cases} \{(c+1)\} \cup depth(childeren(b), a, c+1)\ if\ a \in children(b) \\ depth_p(children(b), a, c+1)\ if\ a \notin children(b)\ and\ children(b) \neq \emptyset \\ \emptyset\ if\ children(b) = \emptyset \end{cases}$$

$$where\ a\ is\ a\ subProduct\ of\ b; and\ c\ is\ zero\ at\ the\ start$$

$$AD_p(a,b) = average(\{x = depth(a,x,0) + depth(b,x,0): where\ x \in CA_{p,t}(a,b)\}$$

$$LA_{pd}(a,b) = \begin{cases} 1\ if\ AD_p(a,b) \leq 2 \\ \dfrac{2}{AD_p(a,b)}\ if\ AD_p(a,b) > 2 \end{cases}$$

Based on the two calculated parameters the likelihood of acquaintance of two products can be calculated as:

$$LA_p(a,b) = \frac{K_1.LA_{pi}(a,b) + K_2 LA_{pd}(a,b)}{K_1 + K_2}$$

$$where\ K_1 and\ K_2\ are\ constants\ with\ the\ default\ value\ of\ one$$

The constants can be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred form the users interaction with the system.

Likelihood of acquaintance of a sub-product with a feature: A sub-product and a feature are acquainted if the feature is a feature of a sub-product that constitutes the sub-product. We can calculate the likelihood using:

$$LA_{fp,p}(x,y) = \frac{|\{p: x \in\ features(p)\ and\ y \in subProducts(p)\ where\ p \in products\ \}|}{|products|}$$

Likelihood of acquaintance of a sub-product with a feature-kind: A sub-product and a feature-kind are acquainted if a feature of the kind of that feature-kind is a feature of a sub-product and that sub-product is composed of the sub-product. We can calculate the likelihood using:

$$LA_{fkp,p}(x,y) = \frac{|\{p: x \in\ PFK(p)\ and\ y \in subProducts(p)\ where\ p \in products\ \}|}{|products|}$$

Likelihood of acquaintance of a sub-product with a class: A sub-product and a class are acquainted if the class is a class of a sub-product and that sub-product is composed of the sub-product. We can calculate the likelihood using:

$$LA_{p,cp}(x,y) = \frac{|\{p: x \in\ subProducts(p)\ and\ y \in classes(p)\ where\ p \in products\ \}|}{|products|}$$

Now based on the likelihoods calculated above we can rank suggestions of sub-products that constitute another sub-product as follows, please note that a is a candidate sub-product:

$$A_1(a) = average(LA_p(a,b_1), \dots, LA_p(a,b_n))$$
$$where\ b_1, \dots, b_n\ are\ candidate\ subProducts\ for\ the\ new\ product,$$

$$A_2(a) = average(\ LA_{fp,p}(c_1,a), \dots, LA_{fp,p}(c_n,a))$$
$$where\ c_1, \dots, c_n\ are\ candidate\ features\ for\ the\ new\ product$$

$$A_3(a) = average(\ LA_{fkp,p}(d_1,a), \dots, LA_{fkp,p}(d_n,a))$$
$$where\ d_1, \dots, d_n\ are\ candidate\ featureKinds\ for\ the\ new\ product$$

$$A_4(a) = average(LA_{p,cp}(a,e_1), \dots, LA_{p,cp}(a,e_n))$$
$$where\ e_1, \dots, e_n\ are\ candidate\ classes\ for\ the\ new\ product$$

$$Rank_{p,sp}(a) = \frac{K_1A_1(a) + K_2A_2(a) + K_3A_3(a) + K_4A_4(a)}{K_1 + K_2 + K_3 + K_4}$$

$$where\ k_1, \dots, k_n\ are\ constant\ and\ are\ one\ by\ default$$

The constants can be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred form the users' interaction with the system.

2. Feature suggestion for a sub-product

Not only should we have reusability for product specifications but also for the building components of the specification. Features are the building blocks of specifications and reusing them enables both reduction in space and also a unified definition for specifications. When specifying a sub-product, each user has different views on what the features of a sub-product are. By providing suggestions for features in the process of sub-product suggestion we speed-up the specification process and guide the user on how he should specify a sub-product.

The ranking for a feature in such suggestion ($Rank_{p,f}(a)$) is based on the likelihood of a feature being in a setting of a sub-product. Other that the likelihood of acquaintance of a sub-product with a feature in the case of composed products, the following Likelihood of acquaintances are also used:

Likelihood of acquaintance for features: Two features are acquainted if they are features of a common sub-product. We can calculate the likelihood of this using the following formula:

$$LA_f(a, b) = \frac{|\{p: a \in features(p) and\ b \in features(p)\ where\ p \in products\}|}{|products|}$$

Likelihood of acquaintance of a class with a feature: A feature and a class are acquainted if the feature is a feature of a sub-product and the sub-product has the class. We can calculate the likelihood using:

$$LA_{fp,cp}(x, y) = \frac{|\{p: x \in\ features(p)\ and\ y \in classes(p)\ where\ p \in products\ \}|}{|products|}$$

Based on the likelihoods we can calculate the ranking for features recommended to the user when adding a new product. The Ranking is calculated by:

$$B_1(a) = average(LA_f(a, b_1), \dots, LA_f(a, b_n))$$

$$where\ b_1, \dots, b_n\ are\ candidate\ features\ for\ the\ new\ product$$

$$B_2(a) = average(\ LA_{fp,p}(a, c_1), \dots, LA_{fp,p}(a, c_n))$$

$$where\ c_1, \dots, c_n\ are\ candidate\ subProducts\ for\ the\ new\ product$$

$$B_3(a) = average(LA_{fp,cp}(a, d_1), \dots, LA_{fp,cp}(a, d_n))$$

$$where\ d_1, \dots, d_n\ are\ candidate\ classes\ for\ the\ new\ product$$

$$Rank_{p,f}(a) = \frac{K_1 B_1(a) + K_2 B_2(a) + K_1 B_3(a)}{K_1 + K_2 + K_3}$$

$$where\ k_1, \dots, k_n\ are\ constants\ and\ are\ one\ by\ default$$

Similar to previous cases the constants can be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred form the users' interaction with the system.

3. Features-kind suggestion for a sub-product

While suggestions of features are suitable to quickly converge the user to a specific specification it might be the case that suggestions provided with that level of details are not useful to the user as he or she is specifying an innovative specification that has not been seen in the system before. As such just recommending feature-kinds that should be covered for the sub-product might come very handy. We can rank existing feature-kinds in the system in order to suggest to the user in the process of specifying a sub-product. This ranking is based on the likelihood of the feature-kind showing up in different settings. Other than the likelihoods discussed before, the following likelihoods of acquaintance come handy in calculating the ranking of the feature-kings for suggestion.

Likelihood of acquaintance for features-kinds: Two feature-kinds are acquainted when they have the following conditions: 1) the two feature-kinds are obligatory features of a common class of sub-products, and 2) the features of with the feature-kinds are present in a common sub-product.

In the first case we can calculate the likelihood using the fallowing formula:

$$LA_{fkc}(x, y) = \frac{|\{a: x \in\ OFK(a)\ and\ y \in OFK(a)\ where\ a \in classes\}|}{|classes|}$$

And for the second case:

$$PFK(p) = \{x: x \in featureKind(t)\ where\ t \in features(p)\}$$

$$LA_{fkp}(x, y) = \frac{|\{p: x \in\ PFK(p)\ and\ y \in PFK(p)\ where\ p \in products\ \}|}{|products|}$$

So based on the two, cases we can calculate the likelihood of acquaintance of two feature-kinds using:

$$LA_{fk}(a, b) = \frac{K_1 . LA_{fkc}(a, b) + K_2 LA_{fkp}(a, b)}{K_1 + K_2}$$

$$where\ K_1 and\ K_2\ are\ constants\ with\ the\ default\ value\ of\ one$$

It is important to point out that the constants here can also be tuned based on the preference a user has on one type of likelihood to another. This can be indirectly inferred form the users interaction with the system.

Likelihood of acquaintance of a class with a feature-kind: A feature-kind and a class are acquainted in two cases: 1) the feature-kind is an obligatory feature-kind of the class, and 2) the feature of the kind of that feature-kind and the class are present in the same product. Obviously the Likelihood of acquaintance in the first case is 1. In other words, if a feature-kind is an obligatory feature-kind of a class then it is always present with the feature-kind.

For the second case we can calculate the likelihood using:

$$LA_{fkp,cp}(x,y) = \frac{|\{p: x \in PFK(p) \text{ and } y \in classes(p) \text{ where } p \in products\}|}{|products|}$$

And so we have:

$$LA_{fk,c}(x,y) = \begin{cases} 1 \text{ if } x \in OFK(y) \\ LA_{fkp,cp}(x,y) \text{ if } x \notin OFK(y) \end{cases}$$

Now based on the likelihoods we can rank suggestions for feature-kinds when the user is specifying a new sub-product. The Ranking is calculated as follows:

$$C_1(a) = average(LA_{fk}(a,b_1), \dots, LA_{fk}(a,b_n))$$

$$where\ b_1, \dots, b_n\ are\ candidate\ featureKinds\ for\ the\ new\ product$$

$$C_2(a) = average(\ LA_{fkp,p}(a,c_1), \dots, LA_{fkp,p}(a,c_n))$$

$$where\ c_1, \dots, c_n\ are\ candidate\ subProducts\ for\ the\ new\ product$$

$$C_3(a) = average(LA_{fk,c}(a,d_1), \dots, LA_{fk,c}(a,d_n))$$

$$where\ d_1, \dots, d_n\ are\ candidate\ classes\ for\ the\ new\ product$$

$$Rank_{p,fk}(a) = \frac{K_1C_1(a) + K_2C_2(a) + K_1C_3(a)}{K_1 + K_2 + K_3}$$

$$where\ k_1, \dots, k_n\ are\ constants\ and\ are\ one\ by\ default$$

Constants here can also be tuned based on user preference.

4. Class suggestion for a sub-product

In the process of guiding the user to specify sub-products, an important part is to suggest the user with existing classes that the sub-product specification could be classified under. This suggestion is an important step due to the fact that users normally are not fully aware of possible options for classes that the product specification might fall under. The ranking for this suggestion is based on the likelihoods for a class to be present in a configuration. Other that the likelihoods defined before, the following likelihood is also needed:

Likelihood of acquaintance for classes: Two classes are acquainted if they are classes of a common product. We can calculate the likelihood of this using the following formula:

$$LA_c(a,b) = \frac{|\{p: a \in classes(p) and\ b \in classes(p)\ where\ p \in products\}|}{|products|}$$

Based on the above likelihood and other previously defined likelihoods, the Ranking for this suggestion is calculated by:

$$C_1(a) = average(LA_c(a,b_1), \dots, LA_c(a,b_n))$$
$$where\ b_1, \dots, b_n\ are\ candidate\ classes\ for\ the\ new\ product$$

$$C_2(a) = average(LA_{p,cp}(d_1,a), \dots, LA_{p,cp}(d_n,a))$$
$$where\ c_1, \dots, c_n\ are\ candidate\ subProducts\ for\ the\ new\ product$$

$$C_3(a) = average(LA_{fp,cp}(d_1,a), \dots, LA_{fp,cp}(d_n,a))$$
$$where\ d_1, \dots, d_n\ are\ candidate\ features\ for\ the\ new\ product$$

$$Rank_{p,c}(a) = \frac{K_1C_1(a) + K_2C_2(a) + K_1C_3(a)}{K_1 + K_2 + K_3}$$
$$where\ k_1, \dots, k_n\ are\ constants\ and\ are\ one\ by\ default$$

Please consider that the constants here can be tuned based on the users preference.

5. Obligatory features-kind suggestion for a class

In order to enhance the user's experience and to guide the user in the specification process, when adding a new class obligatory feature-kinds are recommended to the user. The Ranking for this suggestion is calculated by:

$$Rank_{c,fk} = average(LA_{fk}(a,b_1), \dots, LA_{fk}(a,b_n))$$
$$where\ b_1, \dots, b_n\ are\ candidate\ obligatory\ featureKinds\ for\ the\ class$$

6. Garbage collection and duplicate prevention

When adding new entities, it is important to prevent the user from creating duplicate entities. Capturing duplicates is a difficult task when there are ambiguities. In order to prevent ambiguities for example in names we calculate the similarity of a new entity with existing entitles and if the similarity of the entity to an existing entity is more than a threshold, the system will present a warning to the user and will suggest the existing entity instead of adding a new one. For this the system uses the following similarity functions and the sub-product similarity function ($sim_{fk}, sim_f, sim_p, sim_c$):

Similarity of classes of products: Two classes of products are similar if: a) they have similar names, and b) they have the similar set of obligatory feature-kinds.

The name similarity can be easily calculated using the same equations used for string based features so we can calculate it using the following equations:

$$WS_c(x) = \{\, w\colon w\ is\ a\ word\ used\ in\ name(x)\ \ where\ x\ is\ a\ class\}$$

$$Q_c(a,b) = \{q\colon q = WUP(s_1, s_2); and\ s_1 \in WS_c(a); and\ s_2 \in WS_c(b);\}$$

$$sim_{cn}(a,b) = Max(Q_c(a,b))\ where\ a, b\ are\ classes$$

The obligatory feature-kind similarity of classes are calculated using the following formulas:

$$OFK(x) = \{x\colon x\ is\ Obligatory\ featureKind\ of\ a\}\ where\ x\ is\ a\ class$$

$$sim_{cof}(a,b) = \frac{|OFK(a) \cap OFK(b)|}{|OFK(a) \cup OFK(b)|}$$

So the similarity function of two classes can be calculate as:

$$sim_c(a,b) = \frac{K_1.\, sim_{cn}(a,b) + \ K_2 sim_{cof}(a,b)}{K_1 + K_2}$$

$$where\ K_1\ and\ K_2\ \ are\ constants\ with\ the\ default\ value\ of\ one$$

Where the constants are determined by the users interaction with the system.

Similarity of features-kinds

Two feature kinds are similar when their type is similar (or better to say identical), they have some common units, and their associated features are close. For numerical feature-kinds we should first calculate the average of features associated to a feature-kind that is a vector with the size of the number of units that the feature-kind supports and the value of each tuple is the average of the value of its features that have a specific unit:

$$average_{fk,u}(a,b) = average\ (F_{fk,u}(a,b))$$

$$\text{where } F_{fk,u}\ (a,b) = \{f: f\ is\ featureKind\ a; and\ f\ has\ unit\ b\}$$

and so

$$average_{fk}(a) = \left(average_{fk,u}(a,u_1), \dots, average_{fk,u}(a,u_n)\right)$$

$$where\ a\ is\ a\ featureKind;\ and\ u_1, \dots, u_n\ \in units(a)$$

Finally, the distance of the average of two feature-kinds can be calculated as the Euclidean distance of the average of the two feature-kinds:

$$d_{fk}(a,b) = \sqrt{\sum_{i=1}^{n}(average_{fk,u}(a,u_i) - average_{fk,u}(b,u_i))^2}$$

$$where\ a\ and\ b\ are\ two\ featureKinds;\ and\ u_1, \dots, u_n\ \in units(a) \cap units(b)$$

For string based feature-kinds we calculate the distance between two features-kinds by calculating the distance of all used words in the values of features of that feature-kind to all words used in the values of features of the second feature-kind and then average this value to get to the distance between the feature-kinds:

$$WS_{fk}(x) = \{\ w: w\ is\ a\ word\ used\ in\ the\ value\ of\ a\ feature\ of\ featureKind\ x\}$$

$$d_{fk}(a,b) = Average(Q)$$

$$where\ Q = \{q: q = d_{fk}(s_1, s_2); and\ s_1 \in WS_{fk}(a); and\ s_2 \in WS_{fk}(b);\}$$

Please not that all other feature-kind types can be converted either to numerical or to string.

The distance of the words are calculated based on their “semantic distance” using wordnet. If no “semantic distance” the “Levenshtein distance” is used instead. It is also important to note that when comparing a feature-kind of type string with a numerical feature-kind, the numerical feature-kind is transformed into a string.

Now based on the distance calculated, we can calculate the similarity of two feature-kinds using the following equations:

$$TS_{fk}(fk_1, fk_2) = DT_{fk}\big(type(fk_1), type(fk_2)\big)\ where\ DT_{fk}(A,B)\ is\ a\ predefined\ constant$$

$$US_{fk}(fk_1, fk_2) = \frac{|units(\ fk_1) \cap units(\ fk_2)|}{|units(\ fk_1) \cup units(\ fk_2)|}$$

$$VS_{fk}(fk_1, fk_2) = \frac{1}{|1 - d(fk_1, fk_2)|}$$

$$sim_{fk}(fk_1, fk_2) = \frac{K_1.VS_{fk}(fk_1, fk_2) \ + K_2.TS_{fk}(fk_1, fk_2) + K_3 \ US_{fk}(fk_1, fk_2)}{K_1 + K_2 + K_3}$$

$$where \ K_1, K_3 \ and \ K_3 \ are \ constants \ with \ the \ default \ value \ of \ one$$

Similarity of features: Two features are similar if their feature-kinds and values within a specific unit are similar. For the similarity of two integer features is as follows:

$$sim_f(a, b) = \begin{cases} 0 \ if \ unit(a) \neq unit(b) \\ \frac{1}{|1 - |value(b) - value(b)||} \end{cases}$$

In case of string features the similarity feature is calculated as follows:

$$WS_f(x) = \{ w: w \ is \ a \ word \ used \ in \ value(x) \ \ where \ x \ is \ a \ feature\}$$

$$Q_f(a, b) = \{q: q = sim_{wordnet}(s_1, s_2); and \ s_1 \in WS_f(a); and \ s_2 \in WS_f(b);\}$$

$$sim_f(a, b) = Average\big(Q_f(a, b)\big) \ where \ a, b \ are \ features$$

Please note that if no "wordnet similarity" exists between two words we can replace it with the following formula that is a similarity extracted using the "Levenshtein distance". This also applies to future formulas:

$$sim_{\text{Levenshtein}}(a, b) = \frac{|d_{\text{Levenshtein}}(a, b)|}{\max(length(a), length(b))}$$

Other than similarity one might define a specification that an existing specification is a generalization or details specification of the specification under development. In order for the user to be aware of such cases the system looks into generalization for products and classes and prompts such cases to the user. This is achieved by calculated the likelihood of such cases for the class and product entities (LG_c, LG_p) :

Likelihood of generalization for products

A sub-product is a generalization of another sub-product when: 1) it has a sub-set of features of the original product, 2) it has a sub-set of the constituting sub-product of the original sub-product, and 3) it has a sub-set of the classes of the original sub-product.

The first case, likelihood can be calculated using:

$$LG_{p,f}(a,b) = \begin{cases} \dfrac{|features(a) \cap features(b)|}{|features(b)|} & if\ features(a) \subset features(b) \\ 0\ if\ features(a) \not\subset features(b) \end{cases}$$

For the second case, likelihood can be calculated using:

$$LG_{p,sp}(a,b) = \begin{cases} \dfrac{|subProducts(a) \cap subProducts(b)|}{|subProducts(b)|} & if\ subProducts(a) \\ 0\ if\ subProducts(a) \not\subset subProducts(b) & \\ \subset subProducts(b) & \end{cases}$$

And similarly the third case:

$$LG_{p,c}(a,b) = \begin{cases} \dfrac{|classes(a) \cap classes(b)|}{|classes(b)|} & if\ classes(a) \subset classes(b) \\ 0\ if\ classes(a) \not\subset classes(b) \end{cases}$$

Finally using these criteria we can calculate the overall likelihood as:

$$LG_p(a,b) = \frac{K_1.LG_{p,f}(a,b) + K_2 LG_{p,sp}(a,b) + K_3 LG_{p,c}(a,b)}{K_1 + K_2 + K_3}$$

$$where\ K_1, K_2\ and\ K_3\ are\ constants\ with\ the\ default\ value\ of\ one$$

Likelihood of generalization for classes

A class is a generalization of another class when it has a sub-set of obligatory features-kinds of the original class. We can calculate the likelihood using:

$$LG_c(a,b) = \begin{cases} \dfrac{|OFK(a) \cap OFK(b)|}{|OFK(b)|} & if\ OFK(a) \subset OFK(b) \\ 0\ if\ OFK(a) \not\subset OFK(b) \end{cases}$$

CONSORTIUM

CAS Software AG, Germany

Project coordinator: Dr. Bernhard Koelmel

UNINOVA – Instituto de Desenvolvimento de Novas Tecnologias, Portugal

Technical coordinator: Prof. Luis M. Camarinha-Matos

Universiteit van Amsterdam, Netherlands

iPLON GmbH The Infranet Company, Germany

Steinbeis GmbH & Co., Germany

SKILL Estrategia S.L., Spain

Komix s.r.o., Czech Republic

Prolon Control Systems, Denmark

Member of the:

www.glonet-fines.eu