A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration

Bart Orriens[1], Jian Yang[2], and Mike Papazoglou[1]

[1] Infolab, Tilburg University
PO Box 90153, 5000 LE Tilburg, The Netherlands
{b.orriens,mikep}@uvt.nl
[2] Department of Computing, Macquarie University
Sydney, NSW, 2109, Australia
jian@comp.mq.edu.au

Abstract. Current composite web service development and management solutions, e.g. BPEL, do not cater for flexible and adaptive business collaborations due to their pre-defined and inflexible nature that precludes them accommodating business dynamics. In this paper we propose a rule driven approach for adaptive business collaboration development in which rules drive and govern the development process. We introduce the Business Collaboration Development Framework (BCDF), which provides enterprizes with the context to define their capabilities and business collaboration agreements. Subsequently, we explain how rules can drive and control the business collaboration development process to develop complete, correct and consistent business collaboration agreements that are conform the conditions under which parties wish to cooperate.

1 Introduction

Nowadays enterprizes need to be dynamic and adaptive in order to stay competitive. This has led to an increasing demand for providing business services that can adapt to changes. Recently there has been increasing focus on service oriented computing to deliver flexible and adaptable corporate business services by utilizing existing business services cross organizational boundaries, i.e. via business collaboration. Business collaboration here refers to a cooperation between multiple enterprizes working together to achieve some common business-related goal.

In order to realize this vision enterprizes require an environment in which they can: 1) easily define their business collaboration potential both from a business and technical point of view; and 2) quickly establish the possibility to cooperate with each other. If collaboration is possible, a business collaboration agreement can eventually be negotiated. This type of negotiation also requires that enterprizes can foresee how future changes like new legislative requirements

may influence their ability to cooperate with each other. In addition, enterprizes need to be able to assess how such changes may affect existing collaborations, which moreover need be managed properly (i.e., defined, verified and versioned) and deliver consistent results when executed [30].

Unfortunately, current composite web service development and management solutions including the defacto standard BPEL4WS [11] are too narrowly focused and not capable of addressing the requirements of business collaboration, which relies on agile and dynamic processes. As a result existing technologies and standards to development business collaborations and agreements is very difficult to manage. To address this problem, we propose a rule-based approach where business rules are used to drive and constrain business collaborations. Flexibility then comes from the fact that development of business collaborations is governed by business rules, which further more can be chained and used for making complex decisions and diagnoses. Adaptability can be achieved as changes can be managed with minimum disruption to existing collaborations.

The ideas presented are illustrated using a complex multi-party, insurance claim handling scenario [18]. The example outlines the manner in which a car damage claim is handled by an insurance company (AGFIL). AGFIL cooperates with several contract parties to provide a service that enables efficient claim settlement. The parties involved are Europ Assist, Lee Consulting Services, Garages and Assessors. Europ Assist offers a 24-hour emergency call answering service to policyholders. Lee C.S. coordinates and manages the operation of the emergency service on a day-to-day level on behalf of AGFIL. Garages are responsible for car repair. Assessors conduct the physical inspections of damaged vehicles and agree repair upon figures with the garages.

2 Business Collaboration Development Framework

Before we discuss how enterprizes can use rules to drive the process of coming to a business collaboration agreement, we shall first explore the context of business collaboration to determine the requirements for our approach. In order to capture the context in which business collaboration development takes place, we have developed the Business Collaboration Development Framework (BCDF). This framework provides context by adopting a three dimensional view in order to achieve separation of concern and modularization in the definition of business collaborations. An overview is shown in Fig. 1.

The first dimension is *collaboration aspects* which place emphasis on the different behaviors of an enterprize in business collaboration; where the purpose and target of development varies [15, 24, 29]: 1) before seeking partners to cooperate with an enterprize will first need to capture its **private behavior** in the *internal business process aspect* (like e.g. [1, 8]); 2) Based on its internal behavior the enterprize can then specify its capabilities in its **exposed behavior** (i.e. its externally visible behavior) in the *participant public behavior aspect* (similar to e.g. WSDL [10] and ebXML CPP [16]); 3) Subsequently, the enterprize can start negotiating with other parties to establish a cooperation. If negotiation is

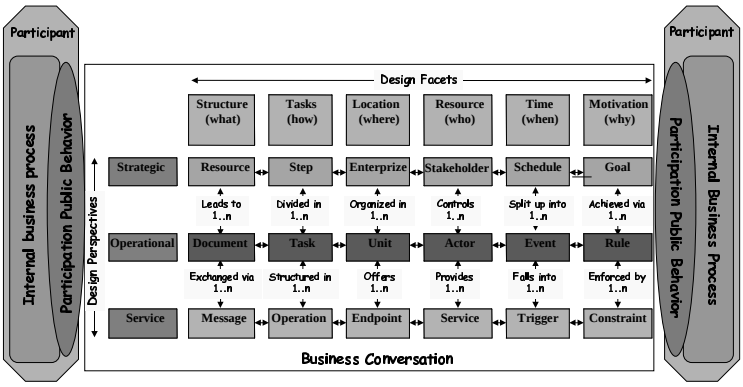

Fig. 1. Business Collaboration Development Framework (BCDF)

successful, the result will be the definition of an **agreed upon behavior** (i.e. the externally observable behavior in a business collaboration) captured in the *business conversation aspect*; where the agreement is based on the participant public behavior aspects of the parties involved (somewhat akin as the merging of two CPPs to form a CPA in the ebXML architecture [16]). Note: the temporal order implied above is illustrative for developing customized, complex business collaborations like the AGFIL scenario. The order may be different when dealing with standardized, simple collaborations (as defined e.g. by RosettaNet [25]).

When enterprizes try to cooperate they must take into consideration both business and technical requirements as well as dependencies between them. This is addressed in the second dimension *levels*, which identifies three different layers of abstraction to allow separation of concern [20, 31]: 1) *strategic level*: at which enterprizes describe their purpose and high level requirements for a business collaboration, with the development process resulting in a strategic agreement expressing shared objectives (like [6, 29]); 2) *operational level*: at which enterprizes depict the operational conditions under which they can cooperate, where the result of development is an operational agreement capturing how set out objectives will be realized in terms of concrete business activities (similar to e.g. RosettaNet [25]); 3) *service level*: at which enterprizes define the technical realization of their business activities, where negotiation amounts to an agreement describing the interactions among the services from the different parties (comparable to e.g. [7], [11]).

At each level of abstraction enterprizes need to consider many issues, for example scheduling, resource usage and logistic optimization at the strategic level. To reduce the complexity resulting from covering all these different issues, the

third dimension of *facets* achieves modularization in the definition of business collaborations. This helps enterprizes describe the different contexts from which a business collaboration can be observed at the different levels. The identified facets are [12, 28, 31]: *what* facet emphasizing the structural view, *how* facet taking functional standpoint, *where* facet expressing geographical facet, *who* facet concerning participants, *when* facet covering temporal aspect, and *why* facet concentrating on rationale. The facets provide a complete coverage for each individual level, where their semantics are dependent on the level that they modularize. Facet interactions reflect the relationships that exist among the different contexts, such as the interaction between the temporal context of a collaboration and its control flow.

In summary, the main conclusion is that business collaboration is highly complex; which in turn makes their development a very complex affair. An important factor that contributes to this complication is that enterprizes must be able to handle a diverse range of changes. In the current business environment changes can occur anywhere, ranging from technological innovation to adoption of new business strategies. Enterprizes need to be able to assess the effect of such changes, both in terms of their potential to collaborate with others as well as regards the consistency of their own behavior. Only in this way, can enterprizes effectively and adequately deal with change in the business collaboration context.

3 Modeling in the BCDF

To capture the three dimensions of collaborations aspects, levels and facets BCDF uses two types of model: meta models and models, both of which are defined for individual levels. Meta models provide design guidelines in terms of classes and their relationships, where depending on the collaboration aspect being modeled additional constraints are placed on the meta-model. Models represent a particular application design, and are derived by populating a meta model's *classes*. Every meta model consists of six classes, where each class captures a particular facet; i.e. for *what*, *how*, *where*, *who*, *when* and *why* facet. Every class constitutes a set of logically related *attributes*. *Associations* connect the classes expressing dependencies among facets. *Mappings* define dependencies among levels by providing links between classes that describe the same facet at different perspectives (illustrated by the arrows between facets at different perspectives in Fig. 1). Dependencies among collaboration aspects are expressed using *connections*, which link the same meta model classes as they are applied in different aspects.

Snippets of exemplary models for the AGFIL application are illustrated in Fig. 2, showing its strategic, operational and service model respectively; where the models are represented based on UML conventions. In order to distinguish different development facets, we represent them in different shapes in their UML models (see also legend in Fig. 2): *what* facet is shown as folded corners, *how* facet as rounded rectangles, *where* facet as plaques, *who* facet as octagons, *when* facet

as hectagons, and *why* facet as diamonds. In the following we briefly discuss the purpose of these models in relation to existing work (see [21] for more details).

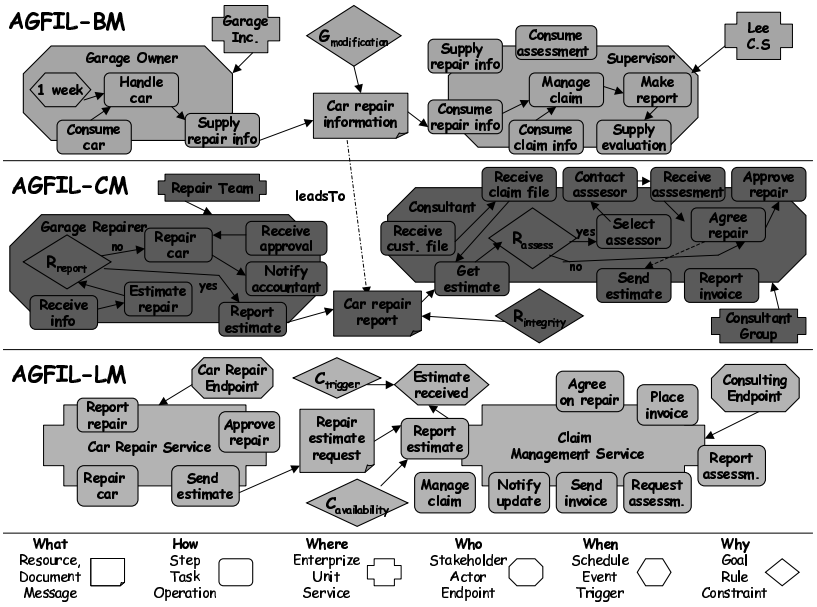

Fig. 2. AGFIL Application Models

At strategic level, strategic models like the AGFIL-BM in Fig. 2 capture purpose and high level requirements of business collaborations, akin to requirements analysis [6, 29]. Participant public behavior aspect (all elements at border of stakeholder like `Lee C.S`) specifies strategic capabilities of individual enterprizes such as `consume car`, whereas internal business process aspect (inside particular stakeholders) identifies the private enterprize processes (e.g. `handle car`) to realize these capabilities. When a strategic agreement is made, business conversation aspect (all modeling elements external to or on boundary of stakeholders like `garage owner`) defines the exchange of resources like `car repair information` between enterprizes to achieve shared strategic objectives.

At operational level, strategic models are concretized in operational models via mappings; where for example `car repair information` in AGFIL-BM `leads to car repair report` in AGFIL-CM (see the dotted arrow in Fig. 2 labeled 'leadsTo'). Note that due to space limitations other mappings are not discussed here. In terms of aspects, in participant public behavior aspect (e.g. elements on border of `garage repairer`) the tasks an actor can perform are depicted e.g. `get estimate` (like ebCPP [16]); whereas internal business process aspect (elements within actor) is similar to e.g. BPML [8] or workflow [1], specifying how and when activities such as `estimate repair` are conducted. Business conversation aspect (all elements on or outside actor borders e.g. `consultant`) captures operational agreements between enterprizes by defining the flow of information between actors; like specified by RosettaNet [25] or BPSS [16].

At service level, operational models are translated in service models where specified activities are realized by services and their operations. Resembling interface behavior in [15], the public participant behavior aspect is captured in models formed by elements placed on the border of individual services like `car repair service` depicting offered operations (akin to e.g. WSDL [10]). Within a service the modeling elements depict internal business process aspect akin to orchestration; where a service internally engages other services to realize its functionality (not shown in Fig. 2). Finally, business conversation aspect (the elements on or outside the border of services) is akin to the notion of choreography [24] defining the agreed upon exchange of messages among services.

4 Using Rules to Drive Business Collaboration Development

We believe that business collaboration development and management is too complex and dynamic to handle manually. We therefore submit that enterprizes need a mechanism which assists them in the flexible development and adaptive management of business collaboration. We adopt a rule driven mechanism for this purpose where the central notion is to let enterprizes explicitly specify the rules under which 1) they conduct their private behavior, 2) are willing to cooperate and 3) are observing in factual business collaborations. These rules can then be used to drive and constrain the process of defining and/or changing a business collaboration agreement. Flexibility comes from the fact that business collaboration development is governed by the rules, which are used for appropriately chaining complex decisions and diagnoses; while adaptability is achieved as changes can be managed with minimum disruption to existing collaborations.

4.1 Types of Rules

Rules in our approach are defined as "precise statements that describe, constrain and control the structure, operations and strategies of a business" [26]. Three main types of rules are employed: development, management and derivation rules. **Development rules** are employed to drive development expressing the peculiarities, originality and values of individual enterprizes. Classified along collaboration aspect in *internal business process aspect* they depict internal guidelines and policies, in *participant public behavior aspect* stipulate cooperations, whereas in *business conversation aspect*, they reflect the conditions agreed upon by the parties involved.

Development rules are also classified along level and facet. Along level they are sub-categorized to enable their usage at different levels of abstraction resulting in a) *strategic rules* expressed in terms of **goals**, b) *operational rules* defined in terms of **business rules**, and c) *service rules* specified in terms of **constraints**. In order to achieve alignment of the different levels in BCDF the strategic, operational and service rules of an enterprize must not contradict each other. Along facet development rules are grouped in relation to the context in

which they are applied, resulting in a) *structural rules* in *what* facet, b) *functional rules* in *how* facet, c) *geographical rules* in *where* facet, d) *participant rules* in *who* facet; and e) *temporal rules* in *when* facet. As the different contexts interact with one another, consistency among these five types of rules is required to define coherent models.

Assurance of outlined forms of consistency is facilitated via **management rules**, serving two purposes: firstly, *consistency rules* ensure semantical soundness of models, i.e. that their meaning is consistent. Consistency rules are subcategorized in: a) *individual rules* dealing with consistency of individual models (e.g. agreement at strategic perspective), b) *alignment rules* dealing with consistency between models at different levels; and c) *compatibility rules* dealing with consistency between models describing different collaboration aspects. Secondly, *completeness rules* and *correctness rules* enforce syntactical soundness; where the former ensure that models and relationships among models are complete, and the latter ensure the correctness of these models and dependencies.

To partially automate the development process we employ so-called *derivation rules*. These rules assist enterprizes by automatically deriving (parts of) models, where they fall in three categories: a) *individual level* enabling derivation of links between elements (i.e. interactions between facets) in strategic, operational and service model, b) *between levels* facilitating derivation of mappings between elements from models at different levels, and c) *between aspects* facilitating derivation of (skeleton) exposed behavior from private behavior as well as (skeleton) agreements from exposed behavior of two parties.

4.2 Rule Specification

Specification of discussed types of development, management and derivation rule is done in the context provided by the meta models and models introduced in Sect. 3. Concretely, rules are grounded on the modeling description atoms (i.e. elements, attributes, links, mappings and connections) that constitute the different BCDF models constraining their existence and/or value. To express rules we adopt RuleML [27], an XML based standard for rule specification currently under development. For conciseness we use its shorthand counterpart POSL [4] (Positional-Slotted Language) here to express rules; whereas RuleML can be used for communication with other parties and execution purposes.

To exemplify, suppose `Garage Inc` has a strategic security rule with regard to `car repair information` in Fig. 2. Let us assume here that for high repair cost estimates `Garage Inc` will want the estimate to be communicated without it being modifiable. For this purpose `Garage Inc` includes the following goal $G_{modification}$ (where the label 'G' reflects the fact that it is a goal) in its public participant behavior aspect model:

$G_{modification}$: property(?ModProp modification,true,carRepairInformation) :-
element(carRepairInformation,resource)

In this Prolog like notation $G_{modification}$ states that if there is an element named `carRepairInformation` its property 'modification' must be set to 'true'.

Goals, business rules and constraints are all expressed in an uniform manner. For example, operational rule $BR_{integrity}$ ('BR' indicating that it is a business rule) states that for all `car repair report` files that contain a car repair value greater than $1000, integrity must be guaranteed:

$R_{garageintegrity}$: property(?IntProp integrity,true,carRepairReport) :-
element(?Element carRepairReport,document),
link(?Link has,carRepairReport,carRepairReportValuePart), property(?ValueProp value,?X,carRepairReportValuePart), greaterThan(?X,1000)

We can define the different kinds of management rule in a similar manner. For example, $MAP_{modification}$ states that for all resources that require modification protections, all documents communicated to realize exchange of these resources must use some form of integrity mechanism:

$MR_{modificationMapping}$: mappingConflict(leadsTo,?X,?Y) :- property(integrity,true,?Y) $\wedge$ element(?X,resource) $\wedge$ property(modification,true,?X) $\wedge$ element(?Y,document) $\wedge$ mapping(leadsTo,?X,?Y)

which states that if a resource X is mapped to a document Y, 'modification' is set to true for the resource and 'integrity' to 'false' for the document, a mapping conflict exists. To conclude our discussion on rule specification the derivation rule $CDR_{matchinteractions}$ exemplifies that the rule language can express all types of rule in a singular manner:

element(?Element document, ?ConversationModel) :- conversation(?ConversationModel), element(?Element document, ?ParticipantModelOne), link(?MyLink receives, ?Source, ?Element, ?ParticipantModelOne), link(?MyLink sends, ?Source, ?Element, ?ParticipantModelTwo), Naf(equal(?ParticipantModelOne,?ParticipantModelTwo)), Naf(equal(?ConversationModel,?ParticipantModel)).

where the intuitive purpose behind this rule is to find all matching receive/send pairs concerning communication of documents in the exposed behaviors of two parties in order to derive a skeleton business collaboration agreement.

4.3 Developing Business Collaboration Agreements

In the previous subsections we introduced the different kinds of rule in our approach, and discussed their specification in context of BCDF. Here we shall explain how the development of business collaboration agreements is driven by combining the development, management and derivation rules introduced in Sect. 4. The development of such agreements constitutes the following: 1) take the exposed behaviors of both parties and merge them into a business conversation aspect model using derivation rules; 2) verify the model using consistency

rules; and 3) any detected inconsistencies can then be resolved, where changes are verified against the exposed behaviors of both parties using *compatibility rules*.

To illustrate, let us look at development of operational agreement between `garage repairer` and `consultant`, where their exposed behavior is as depicted in Fig. 2. That is, `garage repairer` can perform `report estimate` and `receive approval`, whereas `consultant` can carry out `get estimate` and `approve repair`. We merge these two behaviors using *compatibility derivation rules* such as $\text{CDR}_{matchinteractions}$ in subsection 4.2 to generate an initial, skeleton-like agreement. Taking document `car repair report` in Fig. 2 as an example, `garage repairer` has a link between this document and task `report estimate` of type 'send'; whereas `consultant` has a link with its task `get estimate`. Application of $\text{CDR}_{matchinteractions}$ will result in finding a matching receive/send pair, i.e. a feasible interaction between the two.

Once the initial model has been established, the development rules of both parties are applied and checked. Assume that $\text{BR}_{garageintegrity}$ from subsection 4.2 is part of `Garage Inc`'s exposed behavior at operational level stating that `Garage Inc.` will send `car repair report` containing a car repair value greater than $1000 using some integrity mechanism. Also assume that `Lee C.S` has adopted a similar rule $\text{BR}_{leecsintegrity}$, however, it expects `car repair report` to be tamper proof if car repair value greater than $500. To detect such inconsistencies *rule consistency checking* is performed using consistency rules like $\text{CR}_{propertyConsistency}$, which states that if there are two properties of the same type belonging to the same element but with different values, they are in conflict:

$\text{CR}_{propertyConsistency}$: propertyValueConflict(?Type, ?Element) :- property(?Prop1 ?Type, ?Value1, ?Element), property(?Prop2 ?Type, ?Value2, ?Element), notEqual(?Value1, ?Value2)

Through negotiation `Garage Inc.` and `Lee C.S.` agree to observe $\text{R}_{leecsintegrity}$. `Garage Inc.` can ensure that it can accommodate this change as follows: firstly, `Garage Inc.` updates its exposed behavior, where affected areas are identified through compatibility rules like $\text{COR}_{propertyCompatibility}$:

$\text{COR}_{propertyCompatibility}$: propertyValueConflict(?Type, ?Element) :- property(?Prop ?Type, ?Value, ?Element, ?Exposed), property(?Prop1 ?Type, ?Value1, ?Element, ?AgreedUpon), notEqual(?Value, ?Value1)

Then, by using similar compatibility rules governing the relation between its private and exposed behavior, `Garage Inc.` can assess the affect and feasibility of the change on its internal business process activities.

One type of rule not discussed so far concerns the alignment of agreements at different levels. To illustrate their usage, suppose that at strategic level `Lee C.S.` has goal $\text{G}_{modification}$ in its exposed behavior, stating that for all elements of type 'resource' named `car repair information`, their property 'modifica-

tion' must be set to 'true':

$G_{modification}$: property(modification,true,carRepairInformation) :-
element(carRepairInformation,resource)

Now, `car repair information` at strategic level `leads to` `car repair report` at operational level. Let us assume that earlier defined $R_{leecsintegrity}$ applies to `car repair report`. Now goal $G_{carRepairInformation}$ requires modification protection for all claims, whereas $R_{leecsintegrity}$ does not mandate integrity until claim value exceeds $1000). To detect the described inconsistency $MR_{modificationMapping}$ in subsection 4.2 can be used.

This works as follows: suppose we have `car repair information` with value $750. Consequently we also have `car repair report` with document part 'value' equal to $750. According to $G_{modification}$ we can conclude that 'modification' is set to 'true'; whereas from $R_{integrity}$ we can conclude that 'integrity' is set to 'false. Based on these conclusions and the fact that `car repair information` leads to `car repair report`, $M_{modificationMapping}$ results in the conclusion of a mapping conflict; as it states that when 'modification' is 'true', 'integrity' must be true for a mapped resource and document.

In the above we have briefly illustrated how rules can assist enterprizes during the development of business collaboration agreements. Whereas the exact types of rules used depend on the behavior being modeled (i.e. private, exposed or agreed upon behavior) the combined usage of development, management and derivation rules remains principally the same; where development rules ensure that models are conform organizational policies, legislations, etceteras, management rules enforce that they are semantically and syntactically correct, and derivation rules partially automate the development process.

5 Related Work

When it comes to service composition and business collaboration in general, most work has focused on development without taking adaptability into too much consideration. Current solutions like BPEL [11] and ebXML BPSS [16] are pre-determined and pre-specified, have narrow applicability and are almost impossible to reuse and manage. The same applies to works from academia like from workflow [1, 5], system development [6, 29] and enterprize modeling [31].

Relevant work in [3] and [19] describe a generic mechanism for defining WS-Policy based policies (e.g. in [14]), but only web service based rule specification is supported. Also, only rules in participant public behavior aspect are considered. [2] describes a way to establish WS-Agreements between service providers and requesters, but business and technical details are mixed. [9] presents a web service management architecture, however, its metrics cannot capture high level business requirements. [32] describes the rule inference framework DYflow, but there is no clear separation between technical and business rules.

In comparison our work provides a systematic way of specifying development rules for business collaboration in the BCDF context. The business collaboration development process is driven by these development rules to capture the different behaviors of enterprizes; where it is constrained by management rules in terms of 1) conformance and consistency of models, 2) alignment of strategic, operational and service models, and 3) compatibility among different models describing private, exposed and agreed upon behavior; and where it is partially automated by derivation rules.

6 Conclusions

Current standards in business collaboration design, due to their pre-defined and inflexible nature, are precluded from accommodating business dynamics. The challenge is thus to provide a solution in which business collaboration development can be done in an flexible and adaptive manner.

In this paper we presented a rule driven approach for business collaboration development. We introduced the Business Collaboration Design Framework (BCDF), which gives context for business collaboration modeling. Subsequently we explained how rules drive, control and further the design process to facilitate flexible and adaptive business collaboration development.

Work for future research will foremost be focused on incorporation of payment, quality of service and security details. A prototype for presented approach is currently under development; where an early, partial implementation has been reported in [22].

References

1. W. van der Aalst et al, Business Process Management: A Survey, *Proceedings of the International Conference on Business Process Management, 2003.*
2. A. Andrieux et al, Web Services Agreement Specification (WS-Agreement), *http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf, June 2004*
3. S. Bajaj et al, Web Services Policy Framework (WS-Policy), *http://www-106.ibm.com/developerworks/library/specification/ws-polfram/, September 2004*
4. Harold Boley, Integrating Positional and Slotted Knowledge on the Semantic Web?, *http://www.cs.unb.ca/ bspencer/cs6795swt/poslintweb-talk-pp4.pdf, September 2004*
5. J. Bowers et al, Workflow from within and without, *Proceedings of the 4th European Conference on CSCW, 1995*
6. P. Bresciani et al, Tropos: An Agent-Oriented Software Development Methodology, *Autonomous Agents and Multi-Agent Sytems, Vol. 8, No. 3, pp. 203-236, 2004*
7. D. Burdett et al, Web Service Conversation Language *http://www.w3.org/TR/ws-chor-model/, March 24, 2004*
8. Business Process Modeling Initiative, Business Process Modeling Language, *http://www.bpmi.org, June 24, 2002*
9. F. Casati et al, Business-Oriented Management of Web Services, *Communications of the ACM, Vol. 46, No. 10, pp. 55-60, 2003*

10. E. Christensen et al, Web Service Description Language, *http://www.w3.org/TR/wsdl, March 15, 2001*
11. F. Curbera et al, Business Process Execution Language for Web Services, *http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/, July 31, 2002*
12. B. Curtis et al, Process Modeling, *Communications of the ACM, Vol. 35, No. 9, pp. 75-90, 1992*
13. W. Deiters et al, Flexibility in Workflow Management: Dimensions and Solutions, *International Journal of Computer Systems Science and Engineering, Vol. 15, No. 5, pp. 303-313, September 2000*
14. G. Della-Libera et al, Web Services Security Policy, *http://www-106.ibm.com/developers/library/ws-secpol/, 2002*
15. R. Dijkman et al, Service-oriented Design: A Multi-viewpoint Approach, *International Journal of Cooperative Information Systems, Vol. 13, No. 4, pp. 337-368, 2004*
16. ebXML, *http://www.ebxml.org*
17. D. Fensel et al, The Web Service Modeling Framework WSMF, *Electronic Commerce Research and Applications, Vol. 1, No. 2, pp. 113-137, 2002*
18. P. Grefen et al, CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises, *International Journal of Computer Systems Science & Engineering, Vol. 15, No. 5, pp. 277-290, 2000*
19. P. Nolan, Understand WS-Policy processing, *http://www-106.ibm.com/developerworks/webservices/library/ws-policy.html, 2004*
20. Object Management Group, Model Driven Architecture, *http://www.omg.org/docs/ormsc/01-07-01.pdf, July 2001*
21. B. Orriens et al, Establishing and Maintaining Compatibility in Service Oriented Business Collaboration, *Proceedings of the 7th International Conference on Electronic Commerce, Xi'an, China, August 2005*
22. B. Orriens et al, ServiceCom: A Tool for Service Composition Reuse and Specialization, *Proceedings of the 4th International Conference on Web Information Systems Engineering, Rome, Italy, 2003*
23. M. Papazoglou et al, Service-Oriented Computing, *Communications of the ACM, Vol. 46, No. 10, pp. 25-28, October 2003*
24. C. Peltz, Web services orchestration: a review of emerging technologies, tools, and standards, *Hewlett Packard White Paper, January 2003*
25. RosettaNet, *http://www.rosettanet.org*
26. R. Ross, Principles of the Business Rule Approach, *Addison-Wesley, 2003*
27. RuleML, *http://www.ruleml.org*
28. A. Scheer, Architecture for Integrated Information Systems - Foundations of Enterprise Modeling, *Springer-Verlag New York, Secaucus, NJ, USA, 1992*
29. P. Traverso et al, Supporting the Negotiation between Global and Local Business Requirements in Service Oriented Development, *Proceedings of the 2d International Conference on Service Oriented Computing, New York, USA, 2004*
30. J. Yang, Web Service Componentization: Towards Service Reuse and Specialization, *Communications of ACM, Vol. 46, No. 10, pp. 35-40, October 2003*
31. J.A. Zachman, A framework for information systems architecture, *IBM Systems Journal, Vol. 26, no. 3, pp. 276-292, 1987*
32. L. Zeng et al, Flexible Composition of Enterprise Web Services, *Electronic Markets - The International Journal of Electronic Commerce and Business Media, Vol. 13, No. 2, pp. 141-152, 2003*