

# Managing Non-functional Properties of Inter-enterprise Business Service Delivery

Toni Ruokolainen and Lea Kutvonen

Department of Computer Science  
P.O. Box 68 (Gustaf Hällströmin katu 2b)  
FI-00014 UNIVERSITY OF HELSINKI  
FINLAND

Toni.Ruokolainen@cs.Helsinki.FI, Lea.Kutvonen@cs.Helsinki.FI

**Abstract.** In inter-enterprise business service collaborations management of non-functional properties has become a fundamental issue, as business management and computing infrastructures are expected to align more closely. Already, service-oriented computing emphasizes dynamic binding between the functional elements participating in service collaborations and late encapsulation of properties on them. In such environments, the non-functional service properties are used a) as selection criteria during service discovery and b) as parts of collaboration contracts and service-level agreements in particular.

This paper contributes to the conceptualisation of non-functional properties in the context of service-oriented computing. The nature of non-functional properties is elaborated by metamodels that formalize the concepts and provide facilities for the management of non-functional properties during design time and run time.

## 1 Introduction

Non-functional properties (NFP) affect the behaviour and semantics of service-oriented systems by declaring requirements and constraints on interaction endpoints, communication infrastructure and behavioural patterns for example. To increase maintainability and reusability of the system and its components, non-functional properties should be orthogonal with respect to the functional elements of the system and with each other. While the set of possible non-functional properties is open and can not be predetermined or enumerated due to their context dependency and evolution of systems, their usage can be disciplined by deliberate NFP management facility. In the context of service-oriented computing, non-functional properties are required to be managed over heterogeneous and evolving systems. Such a framework must cater for loose coupling and late encapsulation of properties over service cooperations, as well as adopt a uniform approach for design and deployment of the NFP.

In this paper we introduce preliminary work on establishing a framework for managing non-functional properties of business service delivery. This paper complements and elaborates the roadmap for the management of non-functional properties introduced in [1]. While [1] takes a holistic approach to the management of non-functional properties in service-oriented inter-enterprise communities, this paper elaborates the role

of non-functional properties at the service level. The management framework introduced is based on a set of metamodels that formalize the role of non-functional properties and their relationships with the functional elements of business service collaborations. Moreover, the metamodels provide for development of design tools and runtime repositories for NFP management, and establish a unified vocabulary for discussing the non-functional properties at the meta-level during the negotiations held in business service collaboration establishment phase. An approach for binding non-functional properties to functional elements based on model-driven engineering and aspect-oriented modelling principles is adopted, similarity to [2]. The proposed approach supports multi-level modelling of non-functional properties and results in a framework that detaches business-level properties from their technological counterparts, thus providing endurance against system evolution and heterogeneity.

The contents of the paper is as follows. First the concepts of service-oriented cooperation and service-level relationships are elaborated in Section 2. After that, the foundational metamodels for describing non-functional properties of service relationships are introduced in Section 3. These metamodels provide the facilities for the management of non-functional properties. Section 4 discusses deployment of service-level non-functional properties, and finally, we draw some conclusions in Section 5.

## 2 Elaborating service relationships

The Pilarcos B2B - middleware [3, 4] provides the technological and conceptual context for this discussion. The Pilarcos framework delivers concepts and technologies for inter-enterprise computing which especially emphasize the autonomy of business collaboration participants, and business service interoperability. Interoperability and loose coupling is attained with utilization of the Pilarcos B2B - middleware services [3] that provide facilities for metainformation management and sharing (e.g. service types [5]), service trading facilities for interoperable service delivery, dynamically negotiable collaboration establishment [6], and contract-based governance of business networks [7].

The Pilarcos B2B-middleware framework [3] is based on the tenets of service-oriented computing (SOC) [8]. In this context a “*service*” is considered as a business-level abstraction that is represented by bilateral relationships between legal entities, such as individuals or organizations, and the computing facilities they own. The relationship between legal entities is formalized in an agreement that designates the entities either with a role of a service provider or service consumer. The service provider is committed to deliver functionality conforming with the corresponding service. The service consumer is expected to use the service in a way that is in line with the service provisioning scenario. A service relationship is realized by a set of interactions taken between the computational components provided by the cooperating legal entities. For realizing the service agreement relationships and corresponding interactions, legal entities manage and administrate *business services*. A business service is a technological manifestation of a conceptual service and can be implemented for example using the Web Services [9] technology.

The commitments and expectations associated with a service relationship are formalized by a *service-level agreement* (SLA) [10, 11] between the two parties. Some-



Roles describe the expected behaviour, rights, obligations, as well as position and relationships with respect to other roles for entities willing to act in a cooperative context. A *Cooperation* includes at least two *Role* specifications with at least one connection specified between them. Each *Role* has a unique name within the cooperation for referring the cooperating entities indirectly through their role names.

The set of *AssignmentRule* elements represents constraints that must be fulfilled by an entity taking the corresponding role. In the metamodel that is illustrated in Figure 1 none of the specific role assignment rules are visible, since they are expected to be set by the specializations of the metamodel. Finally, a role definition includes a set of inter-role connections represented by the *Connection* concept which relates the subject role with one or more other roles.

A cooperation use is modelled using the concept of *CooperationContext* that specifies the bindings between a set of roles and a set of entities taking part in the corresponding type of cooperation. A *CooperationContext* must conform with the specified *Cooperation* type, especially each cooperation role must be bound to an entity and the entities must conform with the assignment rules given for the corresponding roles. Roles are attached to cooperation participants using the *RoleBinding* concept. The role attached to an entity constrains and regulates the behaviour and properties of the participant to suit the requirements of the corresponding form of cooperation.

## 2.2 Metamodel for service relationships

The concept of service relationship is formalized by the metamodel illustrated in Figure 2. A *ServiceRelationship* associates a contractual service binding with a set of interactions, and defines a set of role delegations from interaction roles to legal roles. The concepts of *ContractualService* and *Interaction* are refinements of the *Cooperation* metamodel and represent the contractual service bindings and interactions respectively.

*ContractualService* represents a bilateral relationship between legal entities taking one of two types of roles, namely a role corresponding to a service provider (called *ServiceProvider*) and service consumer (*ServiceConsumer*). The roles are subtypes of a generalization named *ServiceRole*. Consequently, the metamodel constrains the entities taking a *ServiceRole* to be of kind *LegalEntity*. A *ContractualService* description includes exactly two roles and one connection between those roles.

The concept of *Interaction* defines a bilateral cooperation that refines the notion of *Cooperation* by associating a behavioural pattern describing the inter-role activities with the *Connection*-elements (see Figure 1). In addition, an interaction medium is associated with the *Interaction* concept that designates general properties of interaction, such as if the activities are asynchronous or not. The role bindings in *Interaction* metamodels are between interaction roles (represented by a concept named *InteractionRole*) and interaction endpoints. An interaction endpoint is an element of kind *EndpointEntity* that includes a description of the behavioural pattern associated with the corresponding endpoint. Naturally, the behaviour manifested by an endpoint and the behaviour prescribed by the interaction connection have to be compatible; applicable formal methods are needed to formalize such a notion of behavioural compatibility. Session subtyping [14] for example can be used for this purpose.

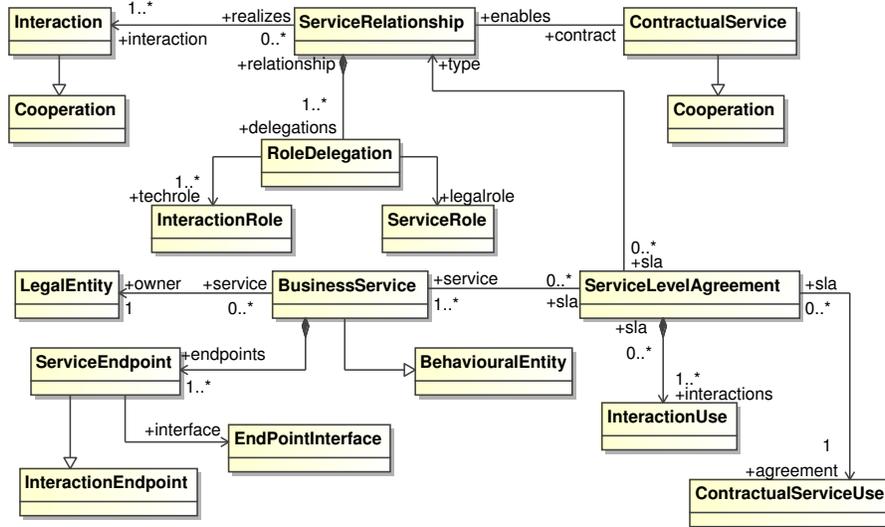


Fig. 2. Service relationship metamodel.

A role delegation represented by the *RoleDelegation* element imposes interaction responsibilities for a legal entity taking part in a service relationship. As an interesting analogue, a *RoleDelegation* can be considered to correspond to a port definition in typical architecture description languages such as Wright [15]: each interaction role corresponds to a port and each connection corresponds somewhat to a connector. In fact, the role delegations are used as part of business role definitions characterizing inter-enterprise collaboration “architectures”; this is however a subject that is not considered further in this paper.

A service relationship is made concrete by a service-level agreement (SLA). An SLA is represented by the concept of *ServiceLevelAgreement* that comprises a set of interaction (*InteractionUse*) and contractual service relationship (*ContractualServiceUse*) instances with corresponding role bindings. A service-level agreement is provided with a set of business services that provide the interaction endpoints and computational resources needed to fulfill the agreement. Each *BusinessService* is considered as a behavioural entity that provides a set of *ServiceEndpoints*. A *BehaviouralEntity* represents entities in a service-oriented computing environment whose existence is motivated by the behavioural properties they own. Such entities are associated with a corresponding *Behaviour* that describes the behavioural patterns the entity is capable of manifesting. A *ServiceEndpoint* is a kind of *InteractionEndpoint* that is provided with a well-defined interface description. A business service is owned by a legal entity whose identity is known such that legal bindings can be accomplished.

The *ServiceLevelAgreement* metamodel comes with a set of constraints some of which are elaborated in the following. First of all, the *InteractionUse* and *Contract-*

*tualServiceUse* elements must conform with the corresponding *Interaction* and *ContractualService* concepts defined in the service relationship that is declared as the type of the *ServiceLevelAgreement*. Moreover, the role bindings and role delegations must match: if a *RoleDelegation* delegates interaction roles  $I_1$  and  $I_2$  to a service role  $S_1$  then the legal entity in role  $S_1$  must also be the provider for a business service that provides the corresponding interaction endpoints. In addition, a few simpler consistency rules are needed in the definition of the *ServiceInteraction* metamodel semantics; these constraints are formalized using the OCL [16] language.

### 3 Non-functional properties of service relationships

Non-functional properties are bound in the Pilarcos framework to service cooperation relationships instead of individual system components. Approaches where NFP:s are bound to individual components are quite often found in traditional QoS research that consider closed, technology-oriented systems. However, we want to emphasize the contextual nature of NFP: non-functional properties are only applicable when expectations are explicitly given also for the co-partner properties and behaviour in corresponding cooperations. NFP can not in general be feasibly defined without specifying the context of its usage. A similar approach to ours has been taken for example in [17], which represents a service model where a service is defined in terms of the interactions and roles involved, and QoS parameters are bound to such a service concept. Non-functional properties are defined as compositions of appropriate aspects. Each non-functional aspect (NFA) gives a well-defined prescription of an orthogonal feature of a cooperation relationship, such as interaction security, in the form of an aspect model [2, 18].

The aspect models in the Pilarcos framework are based on the concept of *NFAType* defined by the metamodel that is illustrated in Figure 3. An *NFAType* defines a kind of a non-functional aspect by declaring its context, the rules how to bind the aspect to the context, and a specification of how to measure or qualify the aspect in the specified context. The context for a non-functional aspect is specified with a reference to a *Cooperation* element.

Following the approach taken in [2, 18] aspects and their binding rules are formalized using graph transformations. An aspect transformation comprises a reference to the base model (a kind of *Cooperation*), and graph patterns (*lhs* for left-hand side and *rhs* for right-hand side) defining the transformation rules. The left-hand side pattern of the aspect transformation is optional; if the *lhs* pattern does not exist, then the *rhs* pattern serves only for the purpose of identifying the relevant elements from the base model to be used in measurements.

A *Measurement* represents a non-functional dimension or feature of a service-oriented cooperation. It prescribes a constrained view on a cooperation (*measures* association) by denoting and naming the entities that are active with respect to the non-functional aspect (*modelview* association). In practice, the measurements are views defined over the metamodels describing the cooperation relationships under investigation. The model view is defined by the right-hand side of the aspect transformation associated with the corresponding *NFAType*. In this regard, the concept of *Measurement* and the approach of considering aspects as views on models is similar to the context models introduced

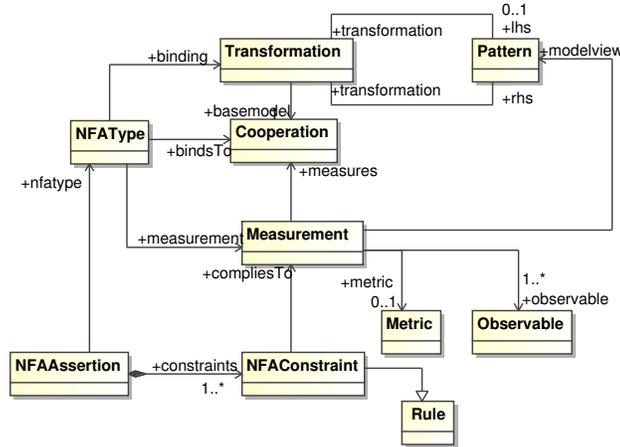


Fig. 3. The metamodel for defining NFA types.

in [19] or feature models as considered in [20, 21]. Measurements prescribe the type of constraints that can be described using the notion of *Metric*. A *Metric* defines a standard for measurements by describing the domain of the measurement (a set of acceptable values) for example, an ordering between values and even functions over the values.

Similarly to [22] we consider non-functional properties as constraints over measurements. Assertions of non-functional aspects are represented by the *NFAAssertion* concept which refers to an *NFAType* and declares constraints over the measurements using the *NFAConstraint* concept. The constraints prescribed must comply with the measurement and the metrics defined in the related *NFAType*. An NFA assertion may for example declare the acceptable choices for interaction encryption from a set of alternative encryption schemes.

An *Observable* represents something that is going to be observed. It exposes knowledge about the properties of a system using well-defined structure and semantics, and can be perceived using some prescribed method of measurement. The notion of observable is needed for monitoring the progress of cooperations, managing the coordination of collaborations, as well as for enforcing so-called service-level agreements (SLAs) [10, 11, 23] and collaboration contracts [7]. The observables are defined by a metamodel that is based on an observer pattern and the ontology introduced in [24]

Non-functional properties of service relationships are defined using the metamodel that is illustrated in Figure 4. A *NonFunctionalProperty* is associated with a kind of *ServiceRelationship*, composes a set of non-functional aspects, and declares additional delegations for roles possibly introduced by the non-functional aspects.

Two different types of non-functional aspects are considered in the definitions of non-functional properties, namely so-called *extra-functional* and *service-contractual* aspects. Extra-functional aspects represented by the *ExtraFunctionalNFAType* concept are bound to service interaction relationships. They affect the functional entities of

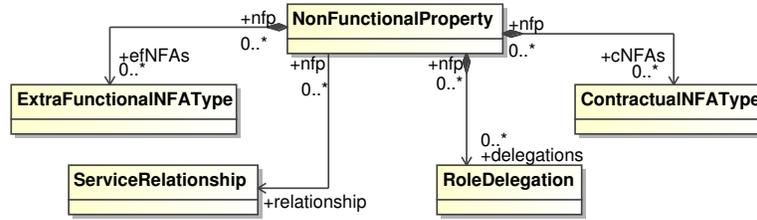


Fig. 4. The metamodel for defining non-functional properties.

service-oriented cooperation by prescribing additional constraints over the behaviour, information, or interaction endpoint properties in the system.

The *service-contractual aspects* represented by the *ContractualNFAType* determine features that are bound to service relationships and affect the legal entities of service-oriented cooperations. Contractual aspects determine expectations, responsibilities and commitments between legal entities that are required for establishing a service relationship with the desired level of quality and performance; they may regulate the timeliness of service invocations, the availability of service implementations, or the level of service quality perceived by an external observer for example.

#### 4 Deploying service-level NFP

The framework described in this paper utilizes a multi-level meta-modelling approach, where the domain models and non-functional aspects and properties are defined using several levels of abstraction (e.g. CIM, PIM, and PSM [25]). When considering non-functional aspects, two different kinds of mechanisms are used for defining the semantics of the aspects. First of all, horizontal model transformations are used for describing the effects of aspect models at the same level of abstraction; such an approach has been proposed for example in [2, 18]. This can be regarded as the “translational semantics” of non-functional aspects. Secondly, a kind of property inheritance mechanism is used for defining semantics of non-functional aspects that require refinement at the lower levels of abstraction. This kind of semantics could be regarded as the “operational semantics” of non-functional aspects, since it defines the meaning and intention of an aspect using some well-defined computing infrastructure (i.e. the abstract service-oriented computing platform) at the lower level of abstraction.

In the following an example case is given which utilizes the mechanisms described above for defining aspect semantics. Model transformations are illustrated informally using UML diagrams [13] and RBML notation [26]. Non-functional properties are annotated to model elements using appropriate mechanisms. In MOF-based modelling languages each *Element* can be associated with *Comment* elements that can embody textual annotations [27]; in this case the annotations declare the names for corresponding *NFAType* declarations.

The example considers an *Interaction* model that is associated with an extra-functional aspect named *securedInteraction* whose intention is to provide confidential interactions by use of a symmetric encryption scheme. A transformation for refining an interaction with respect to the *securedInteraction* aspect is illustrated in Figure 5. It describes a transformation from an *InteractionPattern* described by the *LHS* pattern to a refined one (*RHS*) beginning with key-exchange messages (e.g. using a Diffie-Hellman key-exchange) required by any symmetric encryption scheme. Additional roles of *KeyProvider* and *KeyReceiver* are introduced by the aspect; these roles have to be delegated appropriately in a *NonFunctionalProperty* model using this aspect.

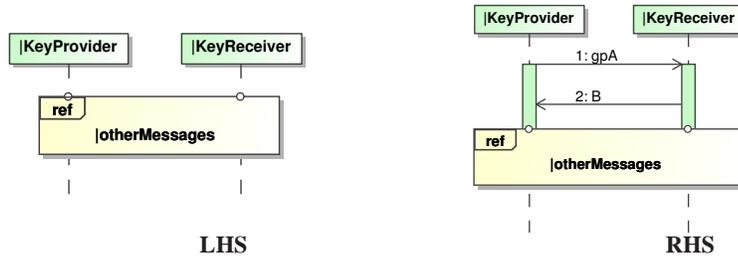
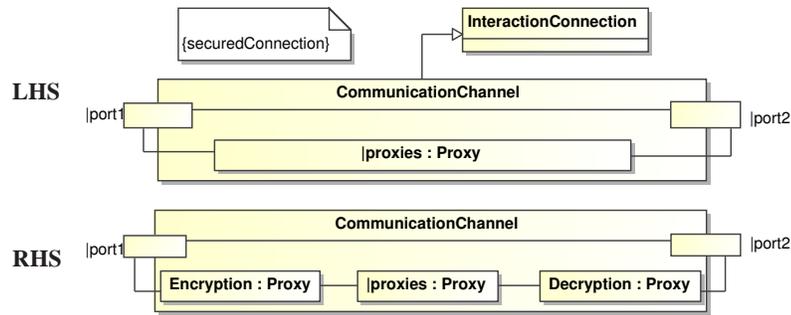


Fig. 5. A transformation introducing key-exchange behaviour to an interaction pattern.

In addition to introducing additional behaviour, the transformation associated with the *securedInteraction* aspect annotates the corresponding *Interaction* element with *securedConnection* declaration. This annotation prescribes a model refinement obligation to be fulfilled at the lower levels of abstraction. The example case considers a *Communication* model (an instance of *Interaction* metamodel) that uses explicit communication channels for realizing interactions. The specific *Communication* model instance inherits the *securedConnection* annotation imposed at the *Interaction* level. In the example case the *securedconnection* NFA has transformational semantics defined by an applicable *NFAType* (especially it is bindable to a *Communication* model) and the corresponding transformation incorporates *Encryption* and *Decryption* proxies to the communication channel as illustrated in Figure 6. If the *securedConnection* aspect did not have a corresponding *NFAType* defined, the annotation would have been propagated further down the abstraction-level.

## 5 Conclusion

This paper introduced a framework for managing non-functional properties of business service delivery. The framework utilizes an approach described in [2] for defining “translational semantics” based on model refinement for aspect models. In addition, a simple property inheritance mechanisms utilizing metamodel element annotations is used for propagating refinement obligations from more abstract aspect models (e.g. CIM-level aspects) to more concrete ones (e.g. PIM-level aspects).



**Fig. 6.** A transformation refining a *CommunicationChannel* with encryption and decryption proxies.

Most importantly, this paper introduced a set of metamodels for enabling a unified design and management of non-functional properties. The metamodels enable development of repositories for sharing the knowledge about different types of non-functional properties in a service-oriented computing environment. Such repositories in par with repositories holding knowledge about the functional elements of business service collaborations (such as service types [5]) are needed for establishing an open and loosely coupled business service trading environment.

The metamodels are founded on a distinction between legal and functional entities, the corresponding cooperation relationships between them, and extra-functional and service-contractual aspects associated to such relationships. Non-functional properties associated with service relationships compose applicable non-functional aspects. The non-functional aspects are kept apart from the actual service relationships they are used in. This enables development of NFA separately, possibly by actors different from the actual business service designers and developers. In such a framework, the business services and non-functional properties can evolve independently.

Domain-specific ontologies and corresponding modelling hierarchies that define and describe the aspects valuable for the corresponding universe of discourse are needed. Especially ontologies and models for observables, possibly recommended by standardization organizations or other communities, are needed before non-functional aspects (especially the service-contractual ones) can be feasibly defined.

## References

1. Kutvonen, L.: Roadmap for the non-functional aspect management in inter-enterprise collaborations. Submitted manuscript. (2007)
2. Köllmann, C., Kutvonen, L., Linington, P., Solberg, A.: An Aspect-Oriented Approach to Manage QoS Dependability Dimensions in Model Driven Development. In: The 3rd International Workshop on Model-Driven Enterprise Information Systems (MDEIS 2007). (2007)
3. Kutvonen, L., Ruokolainen, T., Metso, J.: Interoperability middleware for federated business services in web-Pilarcos. International Journal in Enterprise Information Systems, Special issue on INTEROP-ESA 2005 3 (2007)

4. Kutvonen, L., Metso, J., Ruokolainen, T.: Inter-enterprise collaboration management in dynamic business networks. In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*. Volume 3760 of *Lecture Notes in Computer Science.*, Agia Napa, Cyprus (2005)
5. Ruokolainen, T., Kutvonen, L.: Service Typing in Collaborative Systems. In Doumeingts, G., Müller, J., Morel, G., Vallespir, B., eds.: *Enterprise Interoperability: New Challenges and Approaches*, Springer (2007) 343–354
6. Kutvonen, L., Metso, J., Ruohomaa, S.: From trading to eCommunity population: Responding to social and contractual challenges. In: *Proceedings of the 10th IEEE International EDOC Conference (EDOC 2006)*, Hong Kong (2006)
7. Metso, J., Kutvonen, L.: Managing Virtual Organizations with Contracts. In: *Workshop on Contract Architectures and Languages (CoALa2005)*, Enschede, The Netherlands (2005) To be published.
8. Papazoglou, M.P., Georgakopoulos, D.: Introduction. *Commun. ACM* **46** (2003) 24–28
9. W3C: Web Services Architecture. (2004) W3C Working Group Note.
10. Keller, A., Kar, G., Ludwig, H., Dan, A., Hellerstein, J.: Managing dynamic services: a contract based approach to a conceptual architecture. In: *Network Operations and Management Symposium, IFIP, IEEE* (2002) 513–528
11. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: A service level agreement language for dynamic electronic services. *Electronic Commerce Research* **3** (2003) 43–59
12. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: *Web Services Description Language (WSDL) 1.1. W3C. 1.1 edn.* (2001)
13. Object Management Group: *Unified Modeling Language: Superstructure. 2 edn.* (2005)
14. Vallecillo, A., Vasconcelos, V.T., Ravara, A.: Typing the Behavior of Objects and Components using Session Types. *Electronic Notes in Theoretical Computer Science* **68** (2003) Presented at FOCLASA'02.
15. Allen, R., Garlan, D.: Formalizing architectural connection. In: *ICSE '94*, Los Alamitos, CA, USA, IEEE Computer Society Press (1994) 71–80
16. Object Management Group: *UML 2.0 OCL Specification. 2.0 edn.* (2003) OMG Final Adopted Specification – ptc/03-10-14.
17. Garschhammer, M., Hauck, R., Hegering, H.G., Kempter, B., Radisic, I., Rolle, H., Schmidt, H., Langer, M., Nerb, M.: Towards generic service management concepts a service model based approach. In: *IEEE/IFIP International Symposium on Integrated Network Management.* (2001) 719–732
18. Whittle, J., Araújo, J., Moreira, A.: Composing aspect models with graph transformations. In: *EA '06: Proceedings of the 2006 international workshop on Early aspects at ICSE*, New York, NY, USA, ACM Press (2006) 59–65
19. Röttger, S., Zschaler, S.: Model-Driven Development for Non-functional Properties: Refinement Through Model Transformation. In: *UML 2004. Volume 3273 of Lecture Notes in Computer Science.*, Springer-Verlag (2004) 275–289
20. Czarnecki, K., Kim, C.H.P., Kalleberg, K.T.: Feature Models are Views on Ontologies. In: *SPLC '06: Proceedings of the 10th International on Software Product Line Conference*, Washington, DC, USA, IEEE Computer Society (2006) 41–51
21. Kim, C.H.P.: *On the Relationship between Feature Models and Ontologies.* Master's thesis, University of Waterloo (2006)
22. Zschaler, S.: Towards a semantic framework for non-functional specifications of component-based systems. In: *Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04)*, Washington, DC, USA, IEEE Computer Society (2004) 92–99
23. Skene, J., Skene, A., Crampton, J., Emmerich, W.: The monitorability of service-level agreements for application-service provision. In: *WOSP '07: Proceedings of the 6th international workshop on Software and performance*, New York, NY, USA, ACM Press (2007) 3–14

24. Viroli, M., Moro, G., Omicini, A.: On observation as a coordination paradigm: an ontology and a formal framework. In: SAC '01: Proceedings of the ACM Symposium on Applied Computing, New York, NY, USA, ACM Press (2001) 166–175
25. Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. OMG Press (2003)
26. France, R.B., Kim, D.K., Ghosh, S., Song, E.: A UML-Based Pattern Specification Technique. *IEEE Trans. Softw. Eng.* **30** (2004) 193–206
27. Object Management Group: Meta Object Facility (MOF) Core Specification. 2.0 edn. (2006) OMG Available Specification – formal/06-01-01.