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 re-usability 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 road-map 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

E. Di Nitto and M. Ripeanu (Eds.): ICSOC 2007 Workshops, LNCS 4907, pp. 80–92, 2009.

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. An approach for binding non-functional properties to functional elements based on model-driven engineering and aspect-oriented modelling principles are adopted, similarily 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. Sometimes the SLA can be implicit in nature, that is, the commitments are not explicit but the corresponding business service is provided and used "as is" and as prescribed in the

corresponding service declarations (such as plain WSDL [12] descriptions). The service relationship is associated with a set of bilateral *interaction* relationships between business service *interaction endpoints*.

2.1 Cooperation Metamodel

Service relationships between legal entities and their information systems are based on the concept of *cooperation*, an arrangement between entities taking certain roles in some specified context of joint operation. Two kinds of entities are distinguished in the Pilarcos framework, namely functional and legal entities. Functional entities, such as interaction endpoints and information entities, provide the facilities for delivering the cooperative activities. Legal entities represent the cooperating parties, such as organizations, enterprises and individuals, that are bound by mutual agreements and contracts to deliver the required functionality. The kinds of different entities are defined in a metamodel of their own. The metamodel for cooperation is illustrated in Figure 1 and it essentially prescribes a set of roles and their connection with the cooperating entities.

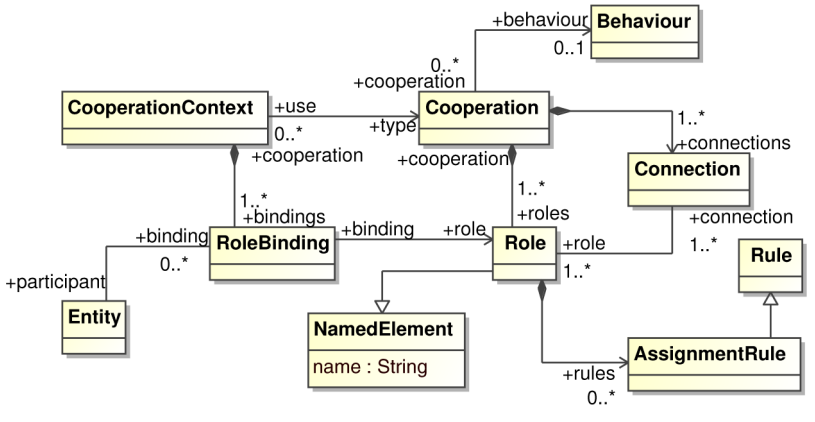

Fig. 1. The cooperation metamodel

A *Cooperation* consists of a set of roles, a set of role inter-connections, and an optional description of the cooperative behaviour. The cooperative behaviour represented by the *Behaviour* concept describes the global behaviour of a cooperation from the "birds-eye" perspective. In this behavioural description the roles are used as actors. For example when using UML sequence diagrams [13] for describing the behaviour, each role in the *Cooperation* is represented by a separate lifeline. Behaviour assigned for roles using the cooperation's *Behaviour* is considered as a prescription of expected role behaviour.

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 interactions 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 bilateral cooperation with a behavioural pattern describing the inter-role activities (see Figure 1). 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.

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,

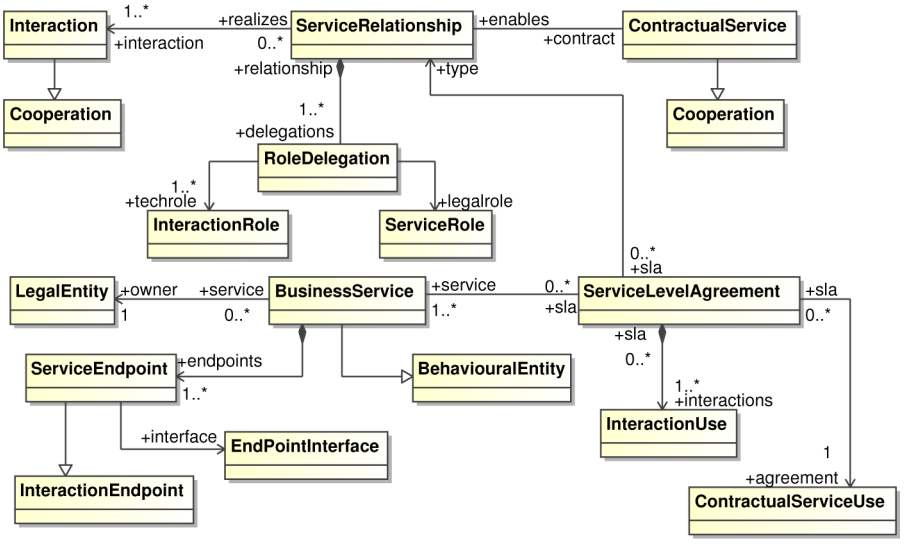

Fig. 2. Service relationship metamodel

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 *ContractualServiceUse* 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.

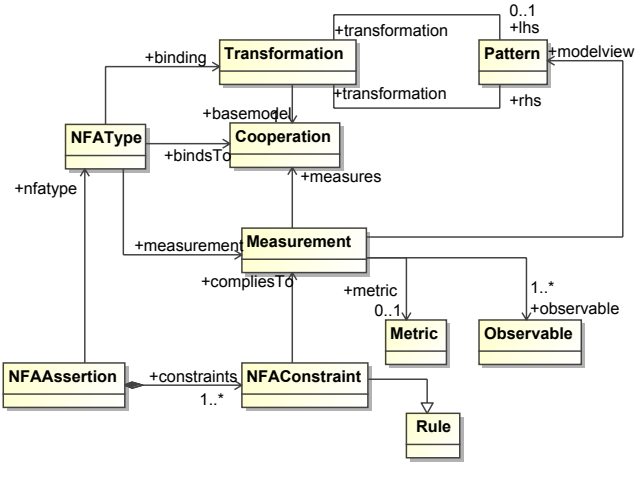

Fig. 3. The metamodel for defining NFA types

Following the approach taken in [18,2] 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 exists, 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 via the *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 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), an ordering between values and even functions over the values, for example.

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 can declare the acceptable choices for interaction encryption from a set of alternative encryption schemes, for example.

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

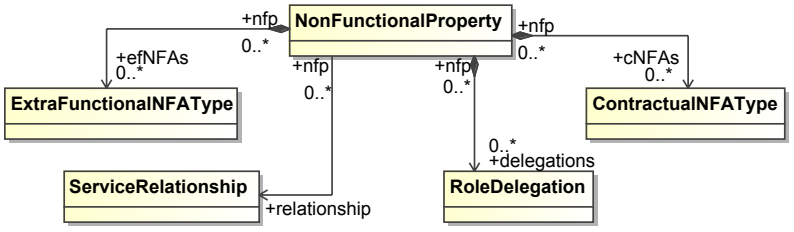

Fig. 4. The metamodel for defining non-functional properties

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 can be 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 [18, 2]. 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

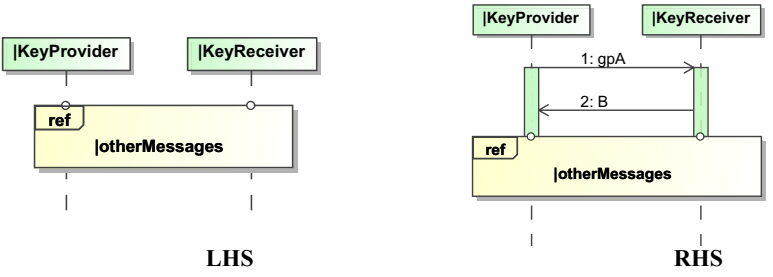

Fig. 5. A transformation introducing key-exchange behaviour to an interaction pattern

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 the have to be delegated appropriately in a *NonFunctionalProperty* model using this aspect.

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.

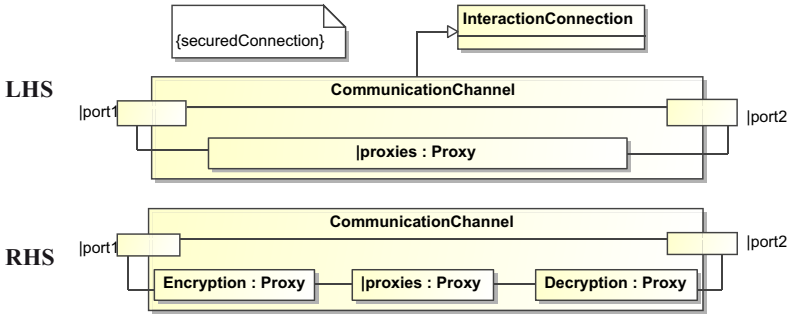

Fig. 6. A transformation refining a *CommunicationChannel* with encryption and decryption proxies

5 Conclusion

While most of the research work in the context of model-driven engineering has been focussing on the functionality of systems, the necessity of incorporating non-functional aspects into model-driven engineering processes has been recognized recently, see for example [28]. In [29] Jonkers et al. introduce a method for integrating functional models with non-functional models in the context of MDA-based service development process. The authors make a distinction between two modelling spaces orthogonal to the MDA asbtraction layers (CIM-PIM-PSM), namely design space and analysis space. Horizontal transformations are used in [29] for propagating information between these two spaces, from design models to formal analysis models. Vertical transformations are used for model refinement within the modelling spaces in traditional MDA-compliant sense [25].

The approach introduced in [29] is also adopted in [30] where the authors introduce their Non-Functional Model Driven Architecture (NFMDA) framework. In the NFMDA approach horizontal transformations are utilized for transforming the design models to non-functional models (CINFM, PINFM, and PSNFM models) attached with notations more applicable for formal analysis, such as Queing Networks [30]. In [2] the authors present an approach for managing several QoS dependability dimensions. The approach applies model-driven development and aspect-oriented techniques for detaching the QoS aspects from software specifications. Graph transformations are applied for weaving the QoS aspects into QoS independent models [2].

This paper contributes to the related work by introducing a set of metamodels for managing non-functional properties in a service-oriented computing environment and relating them to the functional entities. Non-functional properties and functional models of service-based cooperations are glued together by applying principles of model-driven engineering, similarily to [29,30]. The framework presented 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).

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.

Acknowledgement

This work has been performed within the CINCO group (Collaborative and Interoperable Computing Group) at the University of Helsinki, lead by Lea Kutvonen. The group mission is to forward facilities of service interoperability and dynamic business networks. The work is twofold. First, the group has concentrated on a global infrastructure with service discovery and selection, eContracting support, and reputation-based trust management as means to embed social pressure to breach management. Second, the group has started explicating the revolution needs on the software engineering side, through development of the associated service development facilities. The group has worked through a number of national projects funded by the Finnish Funding Agency for Technology and Innovation (TEKES) and Finnish companies, and the INTEROP NoE.

References

1. Kutvonen, L.: Roadmap for the non-functional aspect management in inter-enterprise collaborations (submitted, 2007) (manuscript)
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: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 593–611. Springer, Heidelberg (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, pp. 343–354. Springer, Heidelberg (2007)
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 (CoALa 2005), Enschede, The Netherlands (published, 2005)

8. Papazoglou, M.P., Georgakopoulos, D.: Introduction. Commun. ACM 46, 24–28 (2003)
9. W3C: Web Services Architecture. W3C Working Group Note (2004)
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, pp. 513–528. IEEE, Los Alamitos (2002)
11. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: A service level agreement language for dynamic electronic services. Electronic Commerce Research 3, 43–59 (2003)
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 2002)
15. Allen, R., Garlan, D.: Formalizing architectural connection. In: ICSE 1994, pp. 71–80. IEEE Computer Society Press, Los Alamitos (1994)
16. Object Management Group: UML 2.0 OCL Specification. 2.0 edn., OMG Final Adopted Specification – ptc/03-10-14 (2003)
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, pp. 719–732 (2001)
18. Whittle, J., Araújo, J., Moreira, A.: Composing aspect models with graph transformations. In: EA 2006: Proceedings of the 2006 international workshop on Early aspects at ICSE, pp. 59–65. ACM Press, New York (2006)
19. Röttger, S., Zschaler, S.: Model-driven development for non-functional properties: Refinement through model transformation. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) UML 2004. LNCS, vol. 3273, pp. 275–289. Springer, Heidelberg (2004)
20. Czarnecki, K., Kim, C.H.P., Kalleberg, K.T.: Feature Models are Views on Ontologies. In: SPLC 2006: Proceedings of the 10th International on Software Product Line Conference, Washington, DC, USA, pp. 41–51. IEEE Computer Society, Los Alamitos (2006)
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 2004), Washington, DC, USA, pp. 92–99. IEEE Computer Society, Los Alamitos (2004)
23. Skene, J., Skene, A., Crampton, J., Emmerich, W.: The monitorability of service-level agreements for application-service provision. In: WOSP 2007: Proceedings of the 6th international workshop on Software and performance, pp. 3–14. ACM Press, New York (2007)
24. Viroli, M., Moro, G., Omicini, A.: On observation as a coordination paradigm: an ontology and a formal framework. In: SAC 2001: Proceedings of the ACM Symposium on Applied Computing, pp. 166–175. ACM Press, New York (2001)
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, 193–206 (2004)
27. Object Management Group: Meta Object Facility (MOF) Core Specification. 2.0 edn., OMG Available Specification – formal/06-01-01 (2006)

28. Linington, P.F.: Model Driven Development and Non-functional Aspects. In: WMDD 2004 Workshop, ECOOP 2004, Oslo, Norway, p. 3 (2004)
29. Jonkers, H., Lacob, M.E., Lankhorst, M.M., Strating, P.: Integration and Analysis of Functional and Non-Functional Aspects in Model-Driven E-Service Development. In: Ninth IEEE International EDOC Enterprise Computing Conference (EDOC 2005), pp. 229–238 (2005)
30. Cortellessa, V., Marco, A.D., Inverardi, P.: Non-functional Modeling and Validation in Model-Driven Architecture. In: Working IEEE/IFIP Conference on Software Architecture (WICSA 2007). IEEE, Los Alamitos (2007)