Type management for Service Oriented Computing

Toni Ruokolainen

Dept. of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 UNIVERSITY OF HELSINKI, FINLAND
toni.ruokolainen@cs.helsinki.fi

Abstract. One of the most important infrastructure services of tomorrow's enterprise computing platforms are the services needed for interoperability analysis and management of typing disciplines. This work will study typing disciplines for business collaboration networks that use business services and their compositions to reach their business goals. Research will begin with an analysis of interoperability conflicts inherent in business networks. From these results a typing disciple for business services and networks shall be formulated. Resulting type system is applied in practise as type management functionality in web-Pilarcos project.

1 Introduction

Globalisation of markets, considerably stimulated by the development of ICT technology, introduces new kinds of demands for enterprises willing to stay competitive. Viability of an enterprise is not determined by its core resources or competencies but also by its ability to engage business with various partners, possibly in quite different business domains. An increasing amount of enterprises' income is made inside varying kinds of business networks or as participants in inter-enterprise value and supply chains.

Communication technology has matured enough for a shift from technology driven integration to a more abstract, semantically higher level integration of business entities. Web Services technology and Service Oriented Computing paradigm [1] provides a technological and conceptual framework which can be used to cross the borders between heterogeneous systems. This enables us to concentrate on the foundational problems in enterprise computing environments, such as service reuse, evolution and composition or interoperability of business services.

Services in enterprise computing environments are prone to change and evolution. Enterprises must have the possibility to maintain and adapt their own services while preserving compatibility with business networks they already are participating. Business services offered by enterprises have independent development cycles and lifetimes. An enterprise wants to exploit technological development and refined business processes to retain its competitive edge. To support

these autonomy needs we cannot presuppose that distinct business services will be implemented using same technological platforms or that they would always behave the same.

Concepts and techniques for modeling, establishing and maintaining business networks must reflect the needs stemming from enterprise computing environments. Flexible and dynamic collaborations shall be initiated even for short-term usage and this should be possible without considerable service development or maintenance costs. Tomorrow's enterprise information systems and especially the supporting infrastructure must provide means for efficient establishment and management of collaboration networks in an environment with autonomic and heterogeneous participants. In this context the concept of interoperability becomes prevalent.

This paper is an introduction to the author's research to be conducted during his PhD-studies. Research takes a type based approach for solving interoperability problems stemming from enterprise computing environments and inter-enterprise collaborations. Section 2 discusses preliminary issues in service interoperability. Motivation and an introduction of selected methodology is given in section 3. Research plan is a discussed briefly in section 4.

2 Service interoperability

Three different models for achieving inter-enterprise collaboration can be identified: integrated, unified and federated [2]. In practise only integrated and unified models currently are used. The deficiencies of these two approaches, related to re-usability, flexibility and interoperability of services in inter-enterprise collaborations, make them insufficient models for open systems such as SOC.

In integrated collaboration model all information needed for system interoperability is hard-coded implicitly into the system. This makes reuse of services dedicated to one business network quite impossible in other situations. Use of unified collaboration models, that is relying on common meta-models, is not sufficient either. For example model-driven development, most notably known in form of OMG's MDA approach [3], is a valuable software engineering paradigm and excellent tool for implementing business applications. However it is not necessarily the best option for enterprise integration. Forcing enterprises to strictly obey a common meta-model is not applicable in context of inter-enterprise service integration where flexibility and reuse of business services are embraced.

Because of deficiencies of integrated and unified collaboration models, a federated collaboration model should be used for business service integration. In federated approach no common native meta-model for services is expected. Service communities based on a federated collaboration model require that the information needed for guaranteeing service interoperability is made explicitly available. To provide this information and associated validation mechanisms we need formal descriptions for business services and business networks as well as runtime infrastructure support, such as type management and monitoring facilities [4].

Information contents and requirements for service typing discipline must be extracted using several aspects and viewpoints on interoperability. A thorough analysis of service interoperability in federated communities covering all the necessary aspects and viewpoints is however still missing. What kind of information, what kind of properties it has and what aspects or viewpoints should be covered by the notion of interoperability and related aspects is not evident.

Several definitions and classifications have been addressed regarding interoperability [5–8]. To summarise, interoperability means meaningful co-operation between entities where information is exchanged, it has several faces and it emerges in different levels of abstraction. It is an issue that cross-cuts the whole field of software-engineering, in its all phases and in several levels. Especially this is true in the context of Service Oriented Computing [1].

Service interoperability can be analysed using different approaches. For example different levels of service interoperability (signature, protocol, semantic, quality and context) have been reported in [8]. Another classification has been given in [9], dividing the notion of interoperability into messaging, content, business protocol and policy layers.

Past studies on object and component interoperability [5, 10], and architectural compatibility [11] have been concentrating mainly on the static static aspects of interoperability. However, in enterprise computing environments and inter-enterprise collaborations another level of interoperability aspects emerge. These aspects stem from the properties of enterprise computing environments, namely heterogeneity (freedom of design), autonomy (freedom of action) and dynamism (freedom of configuration) [12]. Above-mentioned properties make static interoperability verification insufficient and also impose requirements on the concepts that are used to formulate the notion of interoperability.

3 Service typing

Service interoperability in federated collaboration model requires a formal characterisation. One formal approach is to use a type based model, where service typing discipline and explicit type management functionality is used to manage service interoperability issues. The role of a service typing discipline is on one hand to guarantee safe use of services (type safety) and on the other hand act as a structuring concept over the universe of services. Type safety is most apparently used during community establishment whereas the type based classification of services is needed during service discovery.

A *service type* defines functional and non-functional properties for a class of business services. Non-functional properties of a service type describe for example QoS-requirements and policies. Functional part of a service type comprises of interface signature (service interface syntax), interface protocol and additionally semantic annotations for exchanged documents (messages). Semantics of service operations in form of pre- and postconditions are not considered relevant in our case, since we consider collaborations as document messaging systems and there are no effects other than the behaviour seen in service and community descrip-

tions. In addition, most types of the both functional and non-functional pre- and postconditions are implicitly available in descriptions as e.g. causal dependencies or non-functional constraints.

The typing discipline for services most importantly addresses the behavioural properties of services: each service type defines its externally visible behaviour as a process. The process description is called an *interface protocol*, stressing the fact that it is not used for process execution and describes only the behaviour of that specific interface. Especially, an interface protocol describes bilateral behaviour. This constraint is made to enhance reuse possibilities of services by increasing the modularity and independence of each interface protocol.

A lot of work has been done on describing the behaviour of services using formal methods such as finite state machines [13], Petri nets [14], process algebras [15]. Use of process algebras for describing and reasoning about services is motivated by such properties as elegant and simple constructs to describe message passing behaviour, natural compositionability of process terms, possibility for operational interpretation for processes, and that they have well-known theories and methods for behavioural reasoning [15–18].

Interface protocol descriptions are based on a process algebra known as π-calculus [17]. Notion of types [19] and possibility to express dynamic, mobile structures [17] are the main motives to choose π-calculus over other process algebras, such as CCS [16]. The notion of session types [20] shall be used to check substitutability (subtyping) and compatibility of interface protocols. Session types express behaviour over communication channels by structuring interrelated communication actions together.

Type repositories are used to enforce verification obligations and to ensure correct use of typing disciplines, in other words type safety. Type repository is a persistent storage of type information. Type information includes descriptions about base types (e.g. integer or string), complex types, service types and relationships between these types. Especially important type relationships for interoperability are those of compatibility (of interface protocols) and substitutability (e.g. subtyping).

Business network model defines the characteristics of an inter-enterprise community, such as topology of the community, processes used between participants (via service types), and community specific functional and non-functional constraints and properties. Especially a business network model composes different service types into business roles and defines connections between the roles. During publication of new business network models, behavioural properties and logical integrity of these descriptions must be verified. Business network model is also the breeding ground for the community contract.

4 Methods and Research Plan

This research will concentrate on interoperability support and management in collaborating communities involving autonomic business services. Work shall start with a categorisation on the nature of interoperability conflicts: what are

their causes and how could they be prevented or observed before they happen. Results of this analysis will be used in formulation of a typing discipline that is used for description of business services and networks.

Typed π-calculus [19, 20] shall be used as theoretical framework for development of business service typing discipline. Use of process algebra provides a formal framework for formulating the notion of interoperability. One of the challenges will be the development of a typing discipline which provides "sufficiently complete" proof of service interoperability, yet supporting independent service and business network evolution.

Service typing discipline must allow evolution and reuse of services in a flexible manner. For this purpose it is necessary to separate aspects of communication and coordination such that they can be analysed separately and their properties and structures changed as independently as possible. Communication and coordination should be treated as orthogonal properties, as much as possible, the first one described in the interfaces of services and the second one described in business network models as "glue" composing distinct service types into business roles.

Composition of service types into roles must be supported by the type system in such a way that verification of composite behaviour remains a relatively light procedure. A type effect system mechanism similar to correspondence assertions introduced in [21] could be used to separate the aspects of communication and coordination in business services and their compositions.

The theoretical framework must support propagation of behavioural effects from the coordination dimension to the communication dimension and vice versa such that correct operation of role compositions can be verified. Theories and mechanisms of typed π-calculus, type and effect systems, and model checking shall be studied for this purpose. Resulting typing discipline shall be tested in practise during this work: a type repository functionality is used as part of web-Pilarcos platform [4] to ensure service interoperability.

References

1. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Commun. ACM **46** (2003) 24–28
2. ISO: ISO 14258 – Concepts and rules for enterprise models. ISO TC184 SC5 WG1. (1999)
3. Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. OMG Press (2003)
4. Kutvonen, L., Ruokolainen, T., Metso, J., Haataja, J.: Interoperability middleware for federated enterprise applications in web-Pilarcos. In: INTEROP-ESA'05. (2005)
5. Konstantas, D.: Object oriented interoperability. In: ECOOP '93 - Object-Oriented Programming: 7th European Conference. Volume 707 of LNCS., Springer-Verlag GmbH (1993) 80–102
6. IEC: TC65/290/DC: Device Profile Guideline. (2002)
7. Vallecillo, A., Troya, J.M., Hernandez, J.: New Issues in Object Interoperability. In J. Malenfant, S. Moisan, A.M., ed.: Object-Oriented Technology: ECOOP 2000

Workshops, Panels, and Posters. Volume 1964 of LNCS., Springer-Verlag GmbH (2000)
8. Fang, J., Hu, S., Han, Y.: A service interoperability assessment model for service composition. In: IEEE International Conference on Services Computing (SCC 2004), IEEE (2004) 153–158
9. Benatallah, B., Nezhad, H.M.: Service oriented computing: Opportunities and challenges. In: Revised Selected Papers of Semantic Web and Databases: Second International Workshop, SWDB 2004. Volume 3372 of LNCS., Springer-Verlag GmbH (2005) 1–8
10. Vallecillo, A., Hernandez, J., Troya, J.M.: Component Interoperability. Technical Report ITI-2000-37, University of Malaga (2000)
11. Allen, R., Garlan, D.: Formalizing architectural connection. In: Proceedings of the 16th international conference on Software engineering, IEEE Computer Society Press (1994) 71–80
12. Singh, M.P., Chopra, A.K., Desai, N., Mallya, A.U.: Protocols for processes: programming in the large for open systems. SIGPLAN Not. **39** (2004) 73–83
13. Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: Automatic composition of e-services that export their behavior. In: Service-Oriented Computing - ICSOC 2003. Volume 2910 of LNCS., Springer-Verlag GmbH (2003) 43–58
14. Hamadi, R., Benatallah, B.: A petri net-based model for web service composition. In: CRPITS'17: Proceedings of the Fourteenth Australasian database conference on Database technologies 2003, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2003) 191–200
15. Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04), Washington, DC, USA, IEEE Computer Society (2004) 43
16. Milner, R.: Communication and concurrency. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1989)
17. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, part I/II. Journal of Information and Computation **100** (1992) 1–77
18. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8** (1986) 244–263
19. Kobayashi, N.: Type Systems for Concurrent Programs. In: Formal Methods at the Crossroads: From Panacea to Foundational Support. Volume 2757 of LNCS. Springer-Verlag (2003)
20. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: Proceedings of the 7th European Symposium on Programming, Springer-Verlag (1998) 122–138
21. Bonelli, E., Compagnoni, A., Gunter, E.: Correspondence assertions for process synchronization in concurrent communications. In: FOCLASA 2003. Volume 97 of Electronic Notes in Theoretical Computer Science., Elsevier (2004) 175–195