A Blueprint of Service Engineering[1]

Christian Zirpins, Toby Baier, Winfried Lamersdorf
Distributed Systems Group- VSIS
University of Hamburg, Hamburg (Germany)
[zirpins, baier, lamersd]@informatik.uni-hamburg.de

Abstract

The rationale behind service oriented computing is to lift inter-organisational integration on a higher level of effectiveness and efficiency. E-services promise to offer means for floating modularisation of arbitrary organisational assets into components that can be dynamically offered, discovered, negotiated, accessed and composed in an open environment. From a technical point of view electronic services are software systems that have to be implemented on top of conventional information and communication technology. As an important step into that direction, the Web Service architecture has laid the foundation for interoperable communication between arbitrary systems. This extended abstract outlines an approach to plan, build and run application-level services on top. Therefore, a fundamental notion of service, originating from distributed systems, is being extended by a specific concept of cooperative interaction processes. Accordingly, an application-level service model and corresponding service engineering mechanisms are proposed that are being realised as middleware based on OGSA Web Services and BPEL4WS processes.

Keywords: Inter-Organisational Integration, Cooperative Interaction Processes, Electronic Services, Workflow, Web-/GRID Service Architecture

1. Introduction: Service Oriented Distributed Applications

Various application domains like electronic business, -government or -education face **recurrent cooperation scenarios**, where a constant change of participants is a predominant characteristic. A typical example is given by business-to-business integration problems [1] that focus on the dynamic relationship between a company and frequently changing partners. Situations like flexible outsourcing of business functions or dynamic supply chain management face similar recurring types of cooperation with interchangeable partners. For example, a company might contract out freight logistics to changing carriers or forwarding agencies. The rationale for this kind of relationship is on the one hand to expose new revenue streams (e.g. providing freight logistics on demand) and on the other to seek for new efficiencies (e.g. outsourcing freight logistics if profitable) both in a form that allows for constant optimisation of partnership settings. Strategic planning of cooperation types, tactical preparation of cooperation settings as well as operational control of functional cooperation are among the main challenges to be tackled here.

In more advanced scenarios, the patterns of functional cooperation are often a subject of variation too, because different partners pose different operational requirements that have to be negotiated between the participants beforehand. For example, a customer cooperates with various carriers that all move goods but impose different procedures of payment. Moreover, when broadening the scope, a party often faces multiple of such cooperative relationships that are in some cases mutually dependent. In order to preserve these dependencies, they have to be made explicit independently from individual partners. For example, a forwarding agency has to ensure that it can move the goods of various individual customers by bringing into action alternating carriers under contract.

Henceforth, **the notion of *service*** will be used to refer to such recurring cooperation scenarios between changing autonomous participants, as it is common in many application domains. In order to substantiate this notion, one can benefit from former work in distributed systems. Revisiting ODP concepts [2], we distinguish the constant class of cooperation (*service type*) from changing cases of cooperation (*service instance*). Service instances can vary in the conditions of cooperation referred to as *service properties* (e.g. QoS) that arise from actual participants. Those participants are typed by *roles*, indicating expected cooperative behaviour within service relationships. Providers offer type and properties of instances they are willing to participate in. Clients observe offers of a specific type, choose a provider with respect to service properties and engage in service instances. Specific interdependencies between services are often referred to as *service composition*. In this case, a participant relates (*composes*) services in which he acts as a provider (*composite services*), to services in which he acts as a client (*service components*), stating how characteristics of the composite service are put down to characteristics of service components.

In terms of characteristics, services on application-level are more complex then those found in classical distributed object systems. Apart from the ‘semantic’ reason (e.g. move goods), the ‘syntactic’ cooperation process (e.g. cus-

[1] We thank the Hewlett Packard Laboratories, Bristol, UK for sponsoring this work.

tomer orders → carrier confirms and ships goods → customer pays) is among the predominant service characteristics. In particular, the focus is on the interaction patterns, that is, the communication processes between roles.

The field of problems faced by organisations in terms of service participation can be structured into strategic, tactical and operational challenges. On strategic level, exposing and expressing semantic and syntactic aspects of service types and their interdependencies requires expressive models and systematic design methodologies (*service modelling*) taking under account the (technological and conceptual) context of participants. For example, a forwarding agency needs models to express a) meaning & procedure of a logistics service it provides b) dependencies of the logistics service to a freight service that it is client for and c) mappings of the service interactions to its internal business information systems. On tactical level, service types have to be constantly maintained to keep track with organisational change (*service type adaptation*). Also on this level, partners have to be located for the types of service a participant is interested in as client (*service discovery*) or provider (*service publication*). On operational level, partners have to be matched (by providers) and chosen (by clients) for service types (*service aggregation*). In some cases, providers additionally have to choose component service types matching the clients of composite services beforehand (*service composition*). During the actual service interaction procedure, terms and conditions of the service have to be ensured (*service coordination & control*). Additional flexibility can be reached by dynamic changes of service instances (*service instance adaptation*).

Generally for all levels, middleware is needed to arrange organisational environments of information- and communication technology (ICT) into a cooperative information system [3] realising services and providing support for the various tasks described above. We refer to such a middleware as *service management system* and to the joint tasks of planning, building and running of *service oriented distributed applications* as *service engineering*.

Current techniques of service oriented computing are strongly focussed on technology. While application-level (i.e. business) service support is out of their scope, they nevertheless pave the way towards it. The emerging Web Service standard [4] provides interoperability between heterogeneous systems by leveraging the expressive power of XML to specify operational interfaces that can be accessed using open internet communication. Thus, organisations can externalize their internal information systems as web enabled components. Those components provide interaction endpoints (subsequently called *ports*) to participate in automated inter-organisational cooperation. Concerning cooperation procedure, the service oriented model adopted by Web Services only defines a very basic type of interaction (i.e. 'broker triangle'). However, web service flow standards like BPEL4WS [5] provide the means for individual definitions of basic interaction processes. This is the crossing point to more general research on cooperative, inter-organisational interaction-processes (e.g. [6-8]) and workflow (e.g. [9-11]), where several practical approaches for application-level services are located (e.g. [12-14]).

The FRESCO project is concerned about **f**oundational **re**search on **s**ervice **co**mposition [15]. The goal is to develop a framework of concepts and technologies that supports organisations in playing the provider role for composite services. As a basis for composition, the focus is on the components first. Subsequently, a fundamental service model was developed that describes basic (i.e. non-composed) application-level services as classes of recurring cooperative interaction. A middleware platform, built on technology of the Web Service family, is currently under development to enable the engineering of such services. In the second part of this extended abstract, a basic blueprint of service engineering is outlined comprising our model & technology platform.

2. A Blueprint of Service Engineering

The FRESCO Service model [16] defines a view on services that is provision-oriented and service-centric. Cooperation procedures that constitute atomic, self-contained parts of a service-relationship are exposed by so called *capabilities*. In particular, capabilities represent *purpose*, *interaction logic* and *resulting artefacts* of the cooperation between organisational *roles*. Capabilities adopt the neutral role of the service itself, introducing a level of indirection between other roles. Unlike meta-level protocols, capabilities take the position of a first-class participant (i.e. coordinator) that might be just virtual or practically enforced. A service is made up by a set of such capabilities.

The most important feature of the model is a separation of capabilities in terms of service content and -provision. *Content* reflects the purpose of a service (e.g. moving goods). It is assumed that it arises from specific resources of the provider (e.g. internal processes, knowledge, people, machines, etc.). To represent service content, cooperation procedures, featuring interactions with such resources, are explicitly exposed as meaningful units of content (e.g. transport tracking…) by capabilities referred to as *assets*. Assets are degenerated in the sense that they don't represent cooperative interaction between roles but monologues of the provider (i.e. binding [7]) that have to be *provided* to clients indirectly by other capabilities. Assets are grouped into a *service core* representing the complete content.

Provision addresses procedures that drive a service and make available content (e.g. negotiating terms and conditions, incorporating assets, etc.), whereby control is exclusively and proactive. Service provision capabilities (hence called "capabilities") are grouped around core assets in a layer called *service shell.* Within a shell, capabilities are mutually interrelated and share a common view on roles and provision-relevant information. Interrelations embody the overall behaviour of provision by defining the global interplay of capabilities. A *service* is fully characterised by defining the basic core and, above all, the enabling shell. Our main focus is on the later.

An associated **technology platform** focuses on a) mapping the service notion to the environment of organisational ICT and b) a set of mechanisms facilitating service engineering on top. Technology mapping is based on the concept of virtual *service engines.* Engines provide a layer of abstraction that is assumed to wrap around diversified ICT systems in order to provide a homogeneous platform for service management. Service types are defined as *schemas* with respect to the engine model. Service instances can be run in any environment implementing engines.

The engine model assumes that all organisational ICT resources of any role (e.g. client's ERP, provider's DBMS…), providing ports for service-related interactions, are represented by means of a homogeneous component model. Shell capabilities appear as *glue* between ports that reflects purpose, interaction logic and result. We represent this glue based on workflow and adopt the WfMC reference model [17] for this purpose. Common patterns are prescribed to define capabilities, their structuring and interrelations by means of the workflow language XPDL, together resulting in a service schema. In particular, a capability maps to the schema of a single workflow definition. Ontology-associations define the *purpose* of *interaction logic* that emerges from the flow of interaction activities and *results* in data artefacts. Interaction activities can be defined with a participant (i.e. a role-associated component-port) to express cooperative procedure or with another capability workflow to express capability interrelation. Coherent sets of capability workflows are grouped together into packages with respect to task-oriented engines (e.g. negotiation engine, payment engine). The shell is given as a top-level package, where each engine is abstracted as a component type itself that realises the enclosed capabilities and gets assigned to a specific role. Thus, various coordination concepts can be expressed including centralised scenarios (orchestration) and distributed scenarios.

In brief, a schema specifies a partitioned set of highly interrelated components with precise interaction behaviour, where a subset `A` represents interacting participants and a subset `B` represents and enforces their interaction patterns. Service engineering is about planning, building and running `B` based on `A`.

Our **concept of service engineering** defines basic mechanisms that allow building customized extensions upon. Besides *modelling*, the focused problems comprise *adaptation*, *aggregation* and *coordination*. As services are inherently complex, we anticipate that support will be needed for their design, that is, a graphical *service modelling language* and *-tool*, which help developers in creating service schemas. This is supposed to be the initial step of the service lifecycle, performed by a participant in provider role. A *service schema manager* provides the functionality to process the schema programmatically. Beside storing and retrieving it, adaptation is its vital task. We adopt a rule based approach that provides a precise and systematic way to change schemas automatically. Back in the service lifecycle, the schema is subject of continuous static adaptation until eventually brought to action.

Then, it's the task of a *service aggregation manager* to create a service instance, based on the schema and a mapping of roles to actual participants. The main problem is to allocate resources of the participants according to the components associated to their roles, thereby optimising resource allocation while guaranteeing a constant and consistent flow of service procedures even when schema or participants change during provision. In the beginning, at least the provider is known and resources for an initial engine have to be allocated from him. *Service engines* are components that manage the aggregation and coordination of the capabilities they realise. The crucial problem is for participants to implement the capabilities of an engine while keeping the service context including associations to other engines and a homogeneous view on roles and data. We propose a generic implementation framework that can be parameterised with executable specifications, generated from the schema. When all engines reach a final state the instance expires and the service lifecycle continues with a new round of static schema adaptation.

A vital characteristic lies in the fact that all management mechanisms are first class components themselves. Thus changes can a) be made at provision time and b) arise from capabilities themselves. For example, a capability can lead to dynamic changes of participants (e.g. a new participant is introduced as a result of a brokerage capability) or dynamic schema adaptation (e.g. a payment procedure is changed as the result of a negotiation capability). Note, that this allows extending the service engineering mechanisms by realising them as capabilities.

A prototype is currently under development that implements the outlined concepts. A UML profile for services and an associated UML tool is meant for graphically designing service schemas and generating XPDL specifications. We adopt the Open Grid Services Architecture (OGSA) [18], that builds on Web Services, as our component model. Thus, adaptation, aggregation and engine components get built as GRID services. Adaptation changes XPDL

service schemas by rule based transformations. It also facilitates BPEL4WS translations that are used to parameterise a set of interrelated service engines. Engines drive BPEL4WS processes with GRID components, allocated by aggregation strategies build upon GRID mechanisms.

3. Conclusion

As inter-organisational relationships increase in terms of quantity and quality the need for new classes of distributed applications arises that allow their effective and efficient management. We focus on recurring cooperation scenarios between changing autonomous participants and the associated class of service-oriented distributed applications. While a suitable technological foundation is already in place to connect the participants, this class misses support to plan, build and run solutions as regards a variety of characteristic problems on application-level.

We propose a service model on top of Web Service technology and address a subset of problems within a basic service engineering approach. Our approach applies process theory and workflow concepts to specify, aggregate, enact and adapt services as interaction patterns between distributed resources. In particular, we adopt a homogeneous view on resources, coordination- and engineering mechanisms that allows for a certain degree of introspection and dynamic self adaptation. We believe that this concept is powerful enough to implement complex service scenarios with customized requirements. In future work, we will use the service engineering mechanisms to examine models and mechanisms for service composition that allow relating and connecting the capabilities of composite services to the capabilities of their service components.

Bibliography

[1] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," *The VLDB Journal, (Springer, Online First, April 3, 2003)*, 2003.

[2] ISO/IEC-JTC1/SC21, "Basic Reference Model of Open Distributed Processing -- Part3: Architecture. International Standard," International Organisation for Standardization 10746-3, 1995.

[3] G. D. Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, M. P. Papazoglou, K. Pohl, J. Schmidt, C. Woo, and E. Yu, "Cooperative Information Systems: A Manifesto," in *Cooperative Information System: Trends and Directions*, M. P. Papazoglou and G. Schlageter, Eds.: Academic Press, 1997.

[4] A. Tsalgatidou and T. Pilioura, "An overview of standards and related technology in Web Services," *Distributed and Parallel Databases*, vol. 12, pp. 135-62, 2002.

[5] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language for Web Services, Version 1.0," BEA, IBM, Microsoft 31 July 2002.

[6] K. Baïna, S. Tata, and K. Benali, "A Model for Process Service Interaction," in *Business Process Management International Conference, BPM 2003. Proceedings*, *LNCS 2678*, M. Weske, Ed.: Springer, 2003, pp. 261 ff.

[7] C. Bussler, "Behavior abstraction in semantic B2B integration," in *Conceptual Modeling for New Information Systems Technologies. ER 2001 Workshops. HUMACS, DASWIS, ECOMO, and DAMA. Revised Papers Lecture Notes in Computer Science Vol.2465. 2002*, H. Arisawa et al, Eds.: Springer Verlag, Berlin, Germany, 2002, pp. 377-89.

[8] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker, "Modeling and Composing Service-Based and Reference Process-Based Multi-enterprise Processes," in *Proc CAiSE 2000*, *LNCS 1789*, L. Bergman, Ed.: Springer, 2000, pp. 247-263.

[9] W. M. P. van der Aalst, "Process-oriented architectures for electronic commerce and interorganizational workflow," *Information Systems*, vol. 24, pp. 639-71, 1999.

[10] E. Colombo, C. Francalanci, and B. Pernici, "Modeling Coordination and Control in Cross-Organizational Workflows," in *Proc. CoopIS/DOA/ODBASE 2002*, *LNCS 2519*, R. Meersmann and Z. Tari, Eds.: Springer, 2002, pp. 91 ff.

[11] Q. Chen and M. Hsu, "Inter-Enterprise Collaborative Business Process Management," Softw are Technology Laboratory, HP Laboratories Palo Alto HPL-2000-107, 2000.

[12] M. Mecella, B. Pernici, M. Rossi, and A. Testi, "A Repository of Workflow Components for Cooperative e-Applications," in *Proceedings of the 1st IFIP TC8 Working Conference on E-Commerce/E-Business*: BICE Press, 2001, pp. 73-92.

[13] O. Perrin, F. Wynen, J. Bitcheva, and C. Godart, "A Model to Support Collaborative Work in Virtual Enterprises," in *Business Process Management International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003. Proceedings*, *LNCS 2678*, W. M. P. v. d. Aalst, A. H. M. t. Hofstede, and M. Weske, Eds.: Springer, 2003, pp. p. 104 ff.

[14] F. Casati, M. Sayal, and Ming Chien Shan, "Developing e-services for composing e-services," in *Advanced Information Systems Engineering. 13th International Conference, CAiSE 2001. Proceedings Lecture Notes in Computer Science Vol.2068. 2001*, K. R. Dittrich, A. Geppert, and M. C. Norrie, Eds.: Springer Verlag, Berlin, Germany, 2001, pp. 171-86.

[15] G. Piccinelli, C. Zirpins, and W. Lamersdorf, "The FRESCO Framework: An Overview," in *2003 Symposium on Applications and the Internet Workshops (SAINT 2003 Workshops)*: IEEE Computer Society, 2003, pp. 120-123.

[16] G. Piccinelli, C. Zirpins, and C. Gryce, "An Architectural Model for Electronic Services," in *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2003)*: IEEE Computer Society, 2003.

[17] WfMC, "Workflow Management Coalition," http://www.wfmc.org, 1.5.2003

[18] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum 2002.