A Web-Service Based Architecture for the Inter-Organizational Coordination of Activities

Rainer Schmidt

Department of Computer Science, University of Applied Sciences
Beethovenstraße 1, 73430 Aalen, Germany

Abstract: To support open process networks and virtual enterprises, activities have to be coordinated not only within an organization, but also between organizations. This inter-organizational coordination of activities creates new requirements, not met by existing architectures. Therefore, a new coordination architecture based on web services is developed. It uses a homogeneous and dynamic composition of so called aspect-elements to support the inter-organizational coordination of activities. The coordination architecture provides coordination autonomy and knowledge encapsulation and offers both process evolution and scalability. At the same time, the benefits provided by web services such as service autarchy, service extensibility, service integration and asynchronous service evolution are maintained.

1 Introduction

Today, enterprises must be able to quickly establish partnerships with other enterprises to combine their competencies and to provide complete solutions to the customer. Examples are open process networks [SeDH01] and virtual enterprises. Such partnerships require the coordination of a multitude of activities across organizational boundaries. Furthermore, the participating enterprises may be others than in the partnerships before und may be others than in the next partnership: there is a high fluctuation of partners. This permanent building and dismantling of partnerships is the key feature of the inter-organizational coordination of activities. Therefore, inter-organizational coordination of activities creates new requirements not met by existing coordination architectures. New activities have to be integrated and existing ones abandoned to reflect changing partnerships. Centralized architectures which endanger the autonomy in defining and executing activities are no more feasible. Distributed services to support the activities have to be integrated in a heterogeneous environment. Enactment architectures which require the synchronous evolution of services are no longer suitable, because there is no central control of service evolution.

Web services [W3WS] offer huge potential benefits for the inter-organizational coordination of activities. They facilitate the use of services across organizational boundaries, hiding different implementation infrastructures, application architectures, programming paradigms and languages. Web service brokers facilitate the integration of services not known before. However, there is no coordination architecture which

Schmidt R. (2004).
A Web-Service Based Architecture for the Inter-Organizational Coordination of Activities.
In *Proceedings of the 1st International Workshop on Computer Supported Activity Coordination*, pages 211-226
DOI: 10.5220/0002680002110226
Copyright © SciTePress

fully maintains the advantages of web services and fulfils the requirements of inter-organizational activity coordination. Therefore, this paper will develop such a coordination architecture based on the so-called aspect-element-oriented schema representation and proceeds as follows. In Section 2, the requirements for the inter-organizational coordination of activities are identified. Section 3 identifies the contributions of web services to the inter-organizational coordination of activities. Section 4 is dedicated to the evaluation of related research. In Section 5, the coordination architecture, the so-called aspect-element-oriented schema representation is introduced. It is implemented in Section 6 as a so-called composite application, using web services.

2 Requirements for the coordination of inter-organizational activities

To clarify the new and augmented requirements created by the inter-organizational coordination of activities, the scenario visualized in Figure 1 is introduced. The scenario describes a company for mechanical engineering and a company for the production of electronic control systems which combine their core competencies. They create a virtual enterprise to gain mutual benefits: The mechanical engineering company profits from making their products more "intelligent" by the integration of electronic control systems. The company for the production of electronic control systems gains access to new markets.

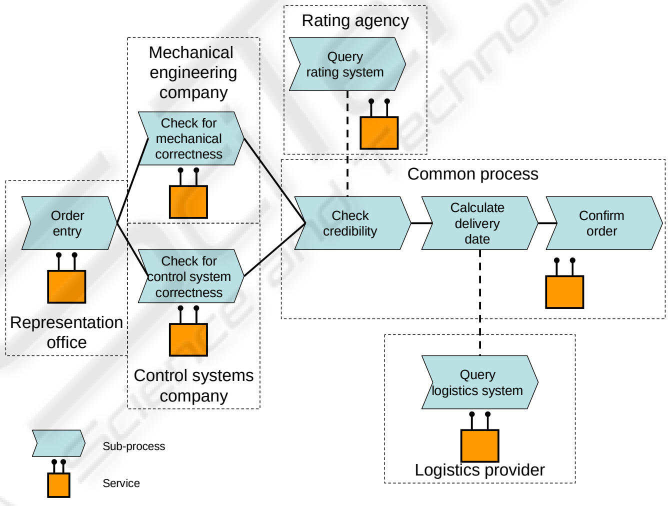

Figure 1: Scenario for the inter-organizational coordination of activities

The activities needed for order processing are shown in Figure 1. Different activities are executed by different enterprises. Services are associated to each activity to support them. These services may reside in other enterprises that the one executing the activity. Examples are the services needed for checking the credibility or calculating the delivery date.

The order processing starts with the order entry. As both companies do not have huge financial resources they rely on representation offices to acquire orders. One of these representation offices is shown. Orders acquired by the representation office are sent to both companies to verify the technical correctness of the order both under mechanical engineering and control systems aspects. Then an external rating agency is consulted to verify the customer's credibility. This is done by querying the rating agencies rating system. The delivery date is calculated by querying the logistics system of a logistics provider, who is responsible for delivery. Finally, an order confirmation is sent out (Possible exceptions such as lacking credibility are covered in a later section). Illustrated by the scenario, the requirements for the inter-organizational coordination of activities can be identified.

- Coordination autonomy

Coordination autonomy is defined as the capability to autonomously coordinate activities. As the participating organizations change quickly, all participants must be able to leave the partnership without loosing the capability to coordinate their activities and they must be able to integrate quickly into a new partnership. Coordination autonomy implies that all participants keep control over their coordination mechanism. Therefore, using a centralized mechanism violates coordination autonomy. If the enterprise owning the centralized coordination mechanism leaves the partnership, the other participants loose their capability to coordinate their activities. Furthermore, partners who do not own coordination mechanisms are forced to stay in the partnership.

In the scenario, coordination autonomy would be lost by using a centralized coordination mechanism installed at the mechanical engineering company. Then, the control systems company is not able to switch to another partner, as it would loose activity coordination.

- Process evolution

The inter-organizational coordination of activities is subject to more changes than the intra-organizational coordination. There are more sources for change requests compared to an isolated environment. These changes can be described as (business) process evolution. It can be differentiated into (business process) model, schema and instance evolution, according to the abstraction layer of the change request.

On the business process model layer the business process model defines a set of elements to represent the activities of the business process and rules how to combine these elements. A business process model can be seen as a kind of language description containing words and a grammar. In most cases the "language" consists of graphical elements and rules for their connection. Changes on the business process model, the so-called business process model evolution, imply the introduction of new modelling elements or rules. They may also imply the integration of new services. The integration of services will be discussed in the section service extensibility and integration.

On the business process schema layer, the "real" business process is formalized as business process schema (or schema for short) using the elements and rules of the business process model. The schema describes the activities of the business process and defines allowed paths for the execution of the activities. It defines the sequence of activities to be executed in the business process and rules for executing activities dependent on preconditions. For example, the schema of an order processing defines the activities to be performed to process the order and rules like “Reject the order if the ordered product is out of production”. The requirement to implement changes on the business process schema layer is called (business process) schema evolution. A typical example of schema evolution is the rearrangement of activities such as executing two activities in parallel which were performed sequentially before.

On the third layer, the business process instance layer, the process schema is used as template for creating business process instances. Business process instances (or instances for short) represent the concrete processing of an order, e.g. the processing of order number 4711. Instance evolution means, that the execution of one or more instances shall deviate from the execution originally specified in the business process schema.

- Knowledge encapsulation

Knowledge encapsulation is necessary to protect the knowledge about proper activity coordination, which is a decisive competitive factor. Due to the short lifetime of partnerships, the knowledge how to coordinate activities has to be kept secret from partners, which may soon participate in competing partnerships. Furthermore, an enterprise may participate in multiple partnerships with companies which are competitors among themselves. Then, these enterprises will attach great importance to the protection of their coordination design.

Knowledge encapsulation is also necessary to make changes manageable. The inter-organizational coordination of activities may have a large extent and involve a multitude of enterprises. Therefore, changes in one company should affect the global process as little as possible: Encapsulation impedes the emergence of cascading change request.

- Scalability

The creation of new partnerships implies the coordination of new and additional activities. Furthermore, in order to react to market changes it must be possible to quickly fulfill an increased demand of products. Therefore an architecture for the inter-organizational coordination of activities must be highly scalable.

- Service autarchy

Service autarchy is defined as the capability to integrate and use independently services. Service autarchy is necessary to cope with business partner fluctuation typical for inter-organizational activity coordination. Only if it is possible to quickly change the provider of a service, a quick adaptation to changing partners is possible. Services autarchy requires the direct access to services without using an intermediary. For examples, if an Enterprise Application Integration (EAI)-system was installed at the mechanical engineering company services provided by the rating agency and the logistics provider could not be directly accessed. This is fatal, if the control systems company tries to switch to another partner. In this case, it would loose access to the services.

- Service extensibility and integration

Above, process evolution has been identified as an important requirement. Process evolution implies that the set of services necessary to execute activities is highly dynamic: New services have to be integrated due to changes to the activities. Therefore, an architecture for inter-organizational activity coordination must be capable of integrating new and services not known before. The services may be highly distributed and heterogeneous because different organizations use different infrastructures. In the scenario, service extensibility and integration provides the possibility to quickly and transparently switch from the service s2 provide by the old logistics provider to the service s3 provided by the new logistics service provider. This service s3 may be implemented in a totally different way as s2.

- Asynchronous service evolution

When coordinating activities across organizations, the underlying services may evolve asynchronously. There is no orchestrated evolution of services. In the scenario, for example, the rating agency could introduce new and enhanced versions of its rating service s1 called s1'. This should not influence the established activities anyway.

In summary, the requirements for the inter-organizational coordination of activities can be divided into two groups. Coordination autonomy, process evolution, knowledge encapsulation and scalability are coordination oriented requirements. The second group are service-oriented requirements: Service autarchy, service extensibility and integration and asynchronous service evolution.

3 Web services and their contributions

Web Services [W3WS], [GrSi02] provide the universal and transparent access to asynchronously evolving services in heterogeneous environments by using near ubiquitous internet technologies such as HTTP [W3C] and XML [XML]. Service requests and responses are encapsulated into XML-documents following the SOAP specification [W3C]. Web services can be described using WSDL [WSDL]. Furthermore, web services provide discovery mechanisms such as UDDI [UDDI] to clients. They allow clients to find services, not known till then. There are coordination related extensions such as the Business Process Execution Language for Web Services (BPEL4WS) [BPEL]. BPEL4WS allows to specify business processes and to define the web services to be used for executing the specified business process [LeRo02]. BPEL4WS is a language definition which provides a set of elements and rules to describe business processes. It can be thought of as business process model, but it does not define an enactment architecture. A design methodology for web services and business processes in general is introduced in [PaYa02]. A detailed framework for services is defined in [BCGL02].

The survey of contributions of web services to the inter-organizational coordination of activities will focus on the service-oriented requirements. Service autarchy is defined as the capability to associate independently services to support the activities. This requirement is fulfilled by web services, because they provide universal access to services. Web services allow integrating services individually and in a peer-to-peer architecture. There is no need for a centralized integration engine such as used in EAI. They also fulfil the requirement of service extensibility and integration because they allow to transparently integrating services across heterogeneous platforms. Furthermore discovery mechanisms such as UDDI allow to integrate up to then unknown services and extent the set of available services. The name space concept of web services, already introduced with XML [XML] establishes a decentralized versioning mechanism: Different versions of a web service can be differentiated by different name spaces. Therefore web services fulfil also the requirement of asynchronous evolution of services.

4 Related research

The CrossFlow project [CROS] developed concepts to support the creation of virtual enterprises by outsourcing activities to other enterprises. The outsourcing is based on contracts which specify the services to be provided. CrossFlow provides an architecture to make and enact these contracts. A virtual market provides mechanisms for matching service requests and offers. Furthermore the configuration of the enactment infrastructure and service monitoring and control is provided. In [AALS2] Message Sequence Charts and Petri Nets are suggested for the specification and verification of inter-organizational coordination of activities. The focus of this approach is to verify the consistence between inter-organizational business process and message sequence charts. In [MPSK98] a dynamic workflow model is developed. Furthermore concepts for the management of web services are proposed and detailed in [MSLH02]. However, no special enactment architecture is proposed. The WISE project [LASS00] developed concepts for the inter-organizational coordination of activities called virtual business processes. Core of the WISE architecture is a centralized engine interpreting the business process schema. It offers also load-balancing, recovery and monitoring mechanisms. The eFlow platform [CaIJ00] provides the composition of services and their adaptation to changes such as the introduction of new services. The dynamic composition of services is addressed in the DySCO approach [PiFW03]. The modeling, composition and execution of services are the aim of the FRISCO project [PiZW03]. Both eFlow, DySCO and FRISCO, however, do not provide an enactment architecture. In [NaLS03] the use of rules and events for activity coordination is proposed. The WorCos concept [Schu99] developed a workflow service, which is integrated into the Object Management Architecture (OMA) [OMG]. Business processes are represented as CORBA objects, which are created by compilation. The objects contain nearly the complete functionality to execute the business process such as role associations, process state and meta data. The IBM Business Process Execution Language for Web Services JavaTM Run Time [BPWS4J] executes business processes using 3 documents. Starting point is the process schema represented in a BPEL4WS document. A WSDL-document [WSDL] specifies the interface of the business process provided to other business processes. A third document is used for specifying the web services used during execution [LeRo02]. BPWS4J creates process instances directly from the process schema by interpretation.

5 The Aspect-element-oriented schema representation

If the approaches listed above are carefully analyzed, two basic architectures can be identified. They are called the direct and the indirect coordination approach. In the direct coordination architecture activities are directly coordinated by a mechanism interpreting the process schemata. No intermediate structure exists as shown in Figure 2. The coordination mechanism also integrates services, for example s1 and s2. High flexibility in changing schemas is the primary advantage of the direct coordination approach. Changes to the process schema can be implemented immediately because the schema is interpreted. For example, a new activity can be easily integrated. The same applies for instance evolution. On the other side, extensions or changes of the business process model are difficult to implement, because the interpretation engine has to be modified. The most serious disadvantage of the direct coordination approach is limited scalability as the interpretation is centralized and cannot be distributed. Because every step in the coordination of activities requires the involvement of the coordination mechanism, a central bottleneck is created. Furthermore, the existence of a centralized coordination mechanism destroys the coordination autonomy and knowledge encapsulation as process definitions are centralized and therefore not under control of the process owner. Service autarchy is lost too, because the centralized coordination mechanism integrates all services used during process execution. The interpretation impedes service extensibility and integration, as the coordination mechanism is not designed in an extensible way. The same applies for asynchronous service evolution.

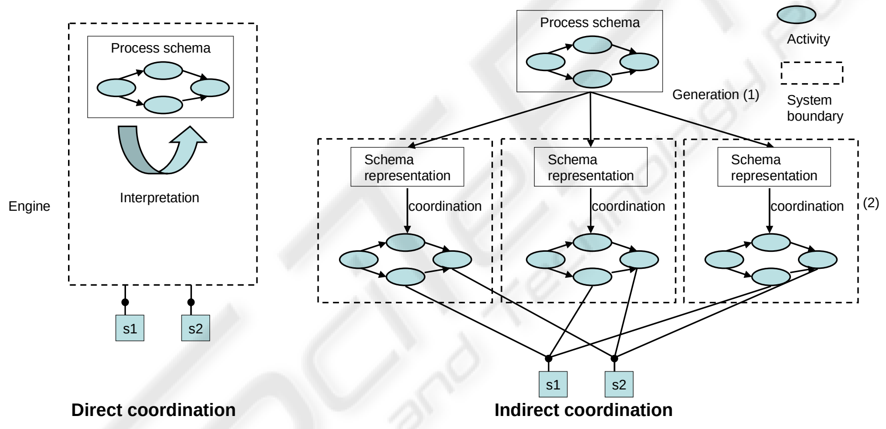

Figure 2: Direct and Indirect coordination

The other basic architecture - indirect coordination - is also shown in Figure 2. Here, coordination is done in two steps. In the first step, a so called schema representation is created (1). In the second step (2), the schema representation coordinates the activities. Thus the coordination of activities can be separated from the creation of the schema representation. Furthermore, services such as s1 and 2 for example have not to be centrally integrated, but can be directly accessed. The use of two steps to generate process instances is the reason for the high scalability of the indirect coordination approach. The creation of the schema representation can be separated from coordination. By copying the schema representation and distributing it, coordination of activities can be done on a multitude of systems. Thus high scalability is achieved. However, the indirect coordination offers only a limited flexibility for schema changes, because they cannot be implemented incrementally: the schema representation is monolithic and static. Coordination autonomy and knowledge encapsulation can be achieved at least by workarounds. The idea is to create separate (sub-) schema representations for each sub process. Service extensibility and integration are impeded because the set of services is fixed and the integration capabilities limited, the same applies for the asynchronous service evolution. Finally the requirement of service autarchy is orthogonal to the concept of the indirect coordination. If a centralized intermediate is used, service autarchy is lost, if a service-oriented architecture is used, service autarchy is maintained. The result of our investigations can be summarized as shown in Figure 3. Standing out is the obvious dilemma between process evolution and scalability. Direct approaches provide schema and instance evolution but limited scalability; indirect approaches provide scalability but no process evolution.

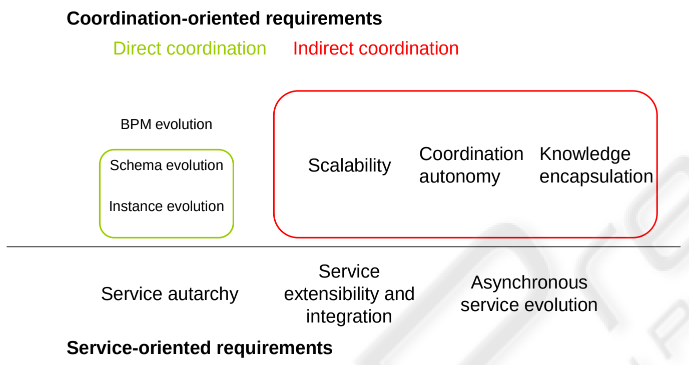

Figure 3: Fulfilment of the requirements for the inter-organizational coordination of activities

Web services fulfil all service-oriented requirements for the inter-organizational coordination of activities. If they are combined with the architectures described so far, they can improve these architectures by providing a better support for service integration. However, the combination of existing architectures with web services is not a remedy for the fundamental deficiencies of these approaches identified above. For example, the direct coordination approach, suffering from a low scalability, does not become scalable if combined with web services. Most important, the dilemma between process evolution and scalability identified in section 4 is can not be solved by the use of web services.

To create an architecture which escapes from the dilemma between process evolution and scalability and which fully uses the benefits offered by web services, two different potential strategies exist. One strategy is to try to improve the scalability of the direct coordination approach. For example, one could use several interpretation engines instead of one. However, this does not solve the fundamental problem that the “flow of control” of the business process always returns to the interpretation engine. The centralized engine remains the bottleneck and therefore, the strategy of improving the direct coordination approach has to be dismissed. The other strategy is to improve the schema representation of the indirect coordination approach in order to support business process model, schema and instance evolution. To do so, two design steps have to be made. First a suitable granularity of the schema elements has to be found. Second, it has to be decided whether a homogeneous or heterogeneous composition of the schema elements should take place. A composition is heterogeneous if there are emphasized elements, for example by coordinating other elements. In a homogeneous composition all elements are handled in the same way.

First ideas concerning an appropriate granularity for activity coordination can be found in [AWBB93] and [LeRo97]. In [AWBB93] the interactions between several activities were identified as cause for lacking evolution support. Thus, interactions between objects were separately handled in so-called interaction patterns. In [LeRo97] the separation of the control flow from the other elements was identified as precondition to avoid applications to become flow dependent and thus inflexible. An analysis why object-oriented software systems in general may become inflexible is the basis of Aspect-Oriented-Programming (AOP) [Kicz96]: Different so-called aspects are mixed together and make the software difficult to adapt. Furthermore, functionality which belongs to one aspect is spread over different classes. Due to this "cross-cutting" functionality [Kicz96] changes to one class imply a multitude of side effects to other classes which participate in the cross-cutting functionality. Therefore, when designing a schema representation, the aspects of business processes have to be identified and separated. Five basic aspects of business processes are identified in [JaBu96]: the functional, control, informational, organisational and operational aspect. The functional aspect describes how a business process is composed of activities. The control aspect describes how activities are executed dependent on the result or completion of other activities. In the organizational aspect, the relation between the business process and the organization structure is established. The operational aspect describes external services to be used during the process. The data flow is covered in the informational aspect. These "core-aspects" of business process are orthogonal dimensions of activity coordination. For example, the organizational structure represented in the organizational aspect may change completely, but the sequence of activities in the process may remain unchanged. However, the approach to encapsulate each aspect into one element does not fulfil the requirements of inter-organizational activity coordination. Scalability is lost because the centralized elements for each aspect become a bottleneck. Furthermore, one has to recall that the activities are distributed to different enterprises and therefore also the services needed to implement them. Thus putting all aspects into one element creates a centralized integration engine comparable to the EAI engine discussed. Based on these considerations, the granularity of the schema representation has to be finer. An appropriate granularity can be achieved when applying aspect elements instead of aspects. Aspect elements are no further dividable, atomic parts of business processes which contain only functionality of one aspect. If aspect elements are applied as granularity for a schema representation, a so-called aspect-element oriented schema representation is created.

The second design step is to decide whether a homogeneous or heterogeneous composition should be chosen. A heterogeneous composition means that one group of aspect elements plays an emphasized role. For example, the elements of the control flow aspect could be put in an emphasized role. The aspect elements of the control flow are directly connected and perform the coordination of all other aspect elements. The problem of heterogeneous composition is that the emphasized aspect elements violate service autarchy by creating "hubs" for accessing the services used for implementing other aspect elements. Contrary to heterogeneous service composition, the homogeneous service composition maintains the service autarchy. There are no aspect elements which are needed to access other aspect elements: all services are accessed directly. Therefore, two kinds of connections between aspect elements are defined. Temporal connections indicate that the aspect elements are executed consecutively. Client-server connections indicate the (multiple) use of a element by another element. To visualize the concept of the homogeneous aspect element oriented schema representation, a small fragment is represented in Figure 4.

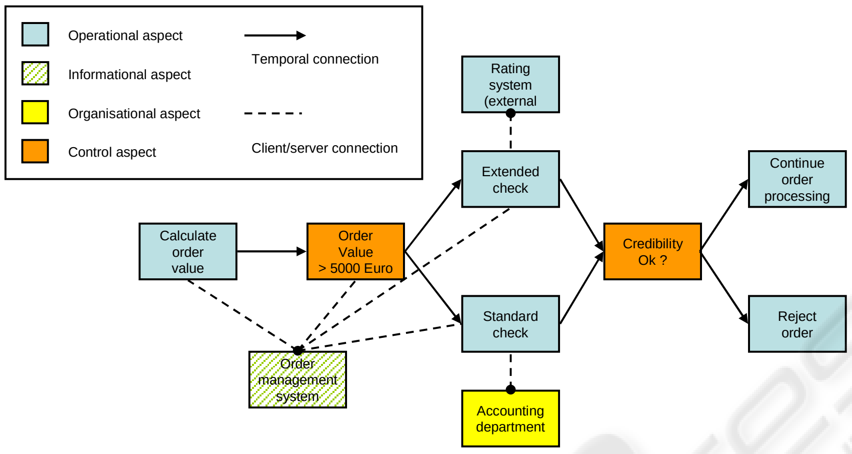

Figure 4: Homogenous, aspect-element-oriented schema representation

The fragment defines, that starting from a calculated order value of 5000 Euro, an extended check of the customer's credibility by querying an (external) rating system shall take place. Otherwise a standard check is done by the accounting department. The representation elements are coloured according to the aspect they belong to. The full lines indicate temporal connections. The dashed lines indicate client-server relationships.

6 Implementing the aspect element oriented schema representation by composite applications

6.1 Composite applications

It can be easily seen, that a dynamic composition is necessary to fulfill the requirements of process model, schema and instance evolution. The elements of the schema representation must be connected in a reconfigurable manner, so that extensions and changes to the schema representation can be implemented dynamically. This is achieved by so-called composite applications on the basis of web services. They break with the thinking that applications and executables have to be one. A composite application is created by a set of interconnected and parameterized services as shown in Figure 5. There is no "executable" any more which contains the application functionality.

The adaptation of web services to an individual composite application is done by specialization, which is done by applying specialization information to the services: the web service is parameterized and connected to other web services depending on the requirements of the composite application. Therefore, the specialization information contains both parameterisation and connection information. The parameterisation information adapts the web service to the individual needs of the composite application. The connection information contains the connections of the web service with other services in the context of the composite application. For different composite applications there is different specialization information. The specialization information is not centrally stored but directly accessible to the web services. The specialization information for different composite applications is discriminated by the so-called global context identifier. The so-called local context identifier differentiates multiple uses of the same service in a composite application. Therefore, every web service "knows" what to do every time.

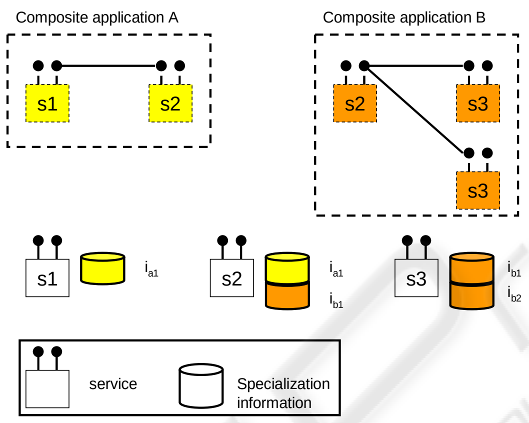

Figure 5: Composite application

An example is given in Figure 5. There are two composite applications A and B. They are created using three web services s1, s2 and s3. Composite application A is created by using web services s1 and s2 with the specialization information denoted with the context identifier i_{a1}. The identifier is composed from a global and a local context identifier. The global context identifier i_a associates the specialization information with the composite application A. The local context identifier denoted "$_1$ " determines the context within composite application A. The specialization s1 contains the connection information representing the connection between s1 and s2. Furthermore s1 and s2 are parameterized. Composite application B is created by using web services s2 and s3 using specialization information denoted with the global context identifier i_b. By evaluation of the global context identifier, web service s2 can differentiate if it is called in the context of composite application A or composite application B. Web service s3 is multiply used within composite application B. Therefore there are two sets of specialization information denoted with the global identifier i_b and differentiated by the local identifiers "$_1$ " and "$_2$ " . If s3 is used for the first time, the specialization information denoted with i_{b1} is used. The second time, the specialization information i_{b2} is used.

6.2 Generating aspect-element oriented composite applications

The process of generating aspect-element oriented composite applications is shown in Figure 6. There are three steps in the generation procedure. Starting point is the specification of the activities to be coordinated, for example a document according to the BPEL4WS specification.

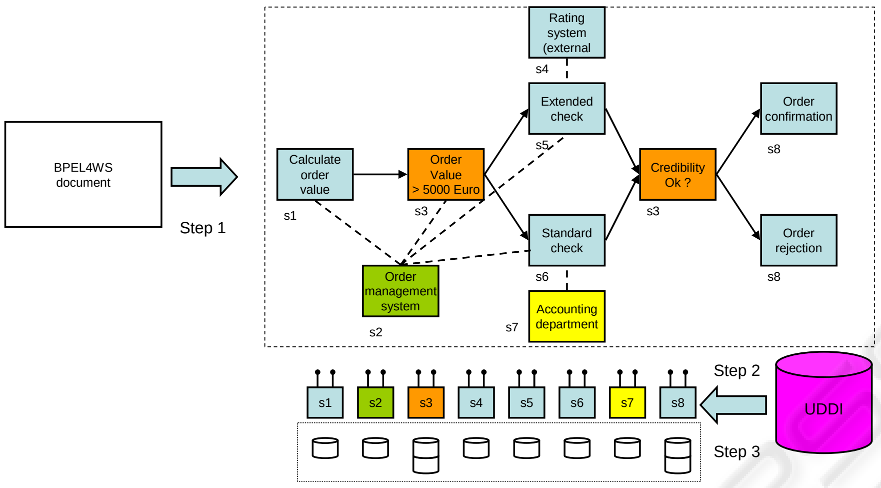

Figure 6: Generation of aspect-element oriented composite applications

1. In the first step the business process specification is analyzed and aspect-element oriented representation elements are identified. That means, the specification is separated into aspects and the aspect-specific content is divided into aspect elements. This procedure has to maintain the connection information of the different aspect elements. The result of the first step is a specification of the business process using aspect elements as granularity.
2. In the second step, for each representation element, an appropriate service has to be found. This may be done by querying a web service repository such as UDDI [UDDI]. In most cases, there is no perfect fit; therefore, the search has to include also web services which can be properly adapted by specialization. Furthermore one web service may be a fit to several representation elements. For example web service s3 can be used to implement both the check whether the order value is greater 5000 euro and the check whether the customer's credibility is ok. This is possible because the web service can be adapted by parameterization to the respective criterion. The first usage of the web service tests the criterion order value, the second test the credibility. The same applies to web service s8 which is used to implement both the order confirmation and the order rejection. The result of step 2 can be seen in Figure 6. A web service is associated to each aspect element.
3. In the third step, appropriate specialization information containing parameterization and connection information is generated and attached to the service. The parameterization information has to adjust the web service to the individual requirements. For example the order value which should initiate a detailed investigation of the customer's liquidity is adjustable. The connection information has to be generated according to the connections of the representation elements. It is important to notice, that the web services s3 and s8 have 2 sets of specialization information because both are used two times. The usages are differentiated by context identifiers, which are issued by the antecedent web service. By this means, s3 will

test the order value if it is called by s1 and test credibility if it is called by s5 or s6.

6.3 Execution of aspect-element oriented composite applications

The execution of aspect element oriented composite applications is done by running iteratively through 4 phases, as shown in **Figure 7**. Each service runs through the phases after it has been initiated by its predecessor. The execution of the composite application stops when the final service is completed.

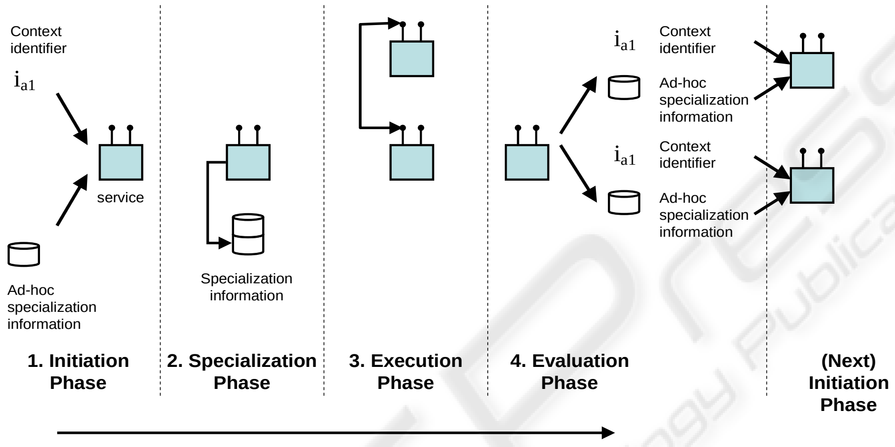

Figure 7: Execution of composite applications

1. The first phase is the initiation phase. The service is started using the information supplied from the calling service. It contains the global and local context identifier. The global identifier is necessary to define the composite application the service is used for. The local identifier defines the context within the composite application. Furthermore, ad-hoc specialization information is provided if necessary. Ad-hoc specialization information overrides or supplements the specialization information attached to the service.
2. The second phase is the specialization phase. The specialization information is selected, depending on the global and local identifier supplied in phase 1. The specialization information is loaded by the service, as it is directly attached directly to it. By applying the specialization information the service is tailored to the individual needs of the composite application. This contains both the parameterization of the service and the connection information to other services.
3. The third phase is the execution phase. In the execution phase, the service is executed using the parameterisation and connection information, which is contained in the specialization information. The client-server connection information is used to identify other services needed during execution.
4. The evaluation phase is the last phase. Here, the results of the service execution are evaluated and the consecutive services are determined using the temporal connection in the connection information. For example, de-

pending on the result of the credibility check, either an acceptance or refusal of the order is initiated.

7 Summary

The inter-organizational coordination of activities is crucial for enterprises which have to quickly adapt or create partnerships with other enterprises to support open process networks and virtual enterprises. In such an environment partners are replaced in existing partnerships and new partnerships are built. This fluctuation of business partners creates new and extended requirements to coordination architectures for activities. The first group are coordination oriented requirements: coordination autonomy, process evolution, knowledge encapsulation and scalability. The second group are service-oriented requirements: Service autarchy, service extensibility, service integration and asynchronous service evolution. The key to fulfil all of these requirements is to use a homogeneous and dynamic aspect-element oriented schema representation as part of an indirect coordination approach. It is implemented by a composite application using specialized web services. The next steps will be the design of a service weaver and the introduction of tracking and tracing mechanisms.

References

[AALS00] W. M. P. van der Aalst: Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. Information Systems, 24(8), 2000

[AWBB93] M. Aksit, K. Wakita, J. Bosch, L. Bergmans, A. Yonezawa: Abstracting Ob-ject Interactions Using Composition Filters. In R. Guerraoui, O. Nier-strasz, M. Riveill (Eds.): Object-Based Distributed Programming, ECOOP '93 Workshop, Kaiserslautern, Germany, July 26-27, 1993. LNCS Vol. 791, Springer Verlag, Berlin, 1994.

[BCGL02] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, M. Mecella: A Foundational Framework for e-Services.

[BPEL] Business Process Execution Language for Web Services, Version 1.0 ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[BPML] BPML.org. Business Process Modelling Language (BPML).

[BPWS4J] http://www.alphaworks.ibm.com/tech/bpws4j

[CaIJ00] F. Casati, S. Ilnicki, Li-Jie Jin: eFlow: a Platform for Developing and Composite e-Services. HP Laboratories Palo Alto. Report HPL-2000-36, March 2000.

[DaHL01] Business Process Coordination: State of the Art, Trends and Open Issues. Proceedings of the 27th VLDB Conference Rom, 2001

[EDI] http://www.unece.org/trade/untdid/welcome.htm

[FDBP01] M.-C. Fauvet, M. Dumas, B. Benatallah, H. Paik. Peer-To-Peer Traced Execution of Composite Services. In Proceedings of the International Workshop on Technologies for E-Services (TES 2001). In cooperation with VLDB 2001.

[GrSi02] S. Graham, S. Simeonov et. Al.: Building Web Services with Java, SAMS Publishing, Indianapolis, Indiana, USA, 2002

[Hare88] D. Harel: On Visual Formalisms. Communications of the ACM, 31(5), Mai 1988, S. 514 – 530.

[J2EE] http://java.sun.com/j2ee/

[JaBu96] S. Jablonski, C. Bußler: Workflow Management - Modeling Concepts, Architecture and Implementation. London 1996

[Kicz96] G. Kiczales: Aspect-oriented programming. ACM Computing Surveys, 28(4), Dezember 1996.

[KlWA99] J. Klingemann, J. Wasch, K. Aberer: Deriving Service Models in Cross-Organizational Workflows. Proceedings of RIDE – Information Technology for Virtual Enterprises, Sydney 1999.

[LASS00] A. Lazcano, G. Alonso, H. Schuldt, C. Schuler: The WISE approach to Electronic Commerce. International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy, Vol. 15, No. 5, September 2000.

[LeRo02] F. Leymann, D. Roller: Business processes in a Web services world: A quick overview of BPEL4WS. ftp://www6.software.ibm.com/software/developer/library/ws-bpelwp.pdf

[LeRo97] F. Leymann, D. Roller: Workflow-based applications. IBM Systems Journal, Volume 36, Number 1, 1997

[MBBN03] B. Medjahed, B. Benatallah, A. Bouguettaya, A. Ngu, A. K. Elmagarmid: Business-to-business interactions: issues and enabling technologies. VLDB Journal 2003, 12: p 59-85

[MKSH02] J. Meng, R. Krithivasan, S. Su and A. Helal, "Flexible Inter-enterprise Workflow Management using E-Services," Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS), Newport Beach, California, USA, June 2002 (pdf).

[MPSK98] J. A. Miller, D. Palaniswami, A. P. Sheth, K. Kochut, H. Singh: WebWork: METEOR2's Web-Based Workflow Management System. Journal of Intel-ligent Information Systems (JIIS), 10(2), 1994, S. 185 — 215.

[MSLH02] J. Meng, S. Y.W. Su, H. Lam and A. Helal, "Achieving Dynamic Inter-organizational Workflow Management by Integrating Business Processes, Events, and Rules," Proceedings of the Thirty-Fifth Hawaii International Conference on System Sciences (HICSS-35), January 2002 (pdf).

[NaLS03] K. Nagarajan, H. Lam, S. Su: Integration of Business Event and Rule Management with the Web Services Model," in the Proceedings of the First International Conference on Web Services(ICWS '03), June 23-26, 2003, Las Vegas, NV.

[OMG] http://www.omg.org

[PaYa02] M. P. Papazoglou, J. Yang: Design Methodology for Web Services and Business Processes. Proceedings of the Technologies for E-Services Third International Workshop, TES 2002, Hong Kong, China, August 23-24, 2002. Editors: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. Lecture Notes on Computer Science, Springer-Verlag, Berlin, Germany. Pages 54 - 64.

[PiFW03] G. Piccinelli, A. Finkelstein, S. L. Williams. "Service-oriented Workflows: the DySCo framework" To appear: Proc. Euromicro Conference, Antalya, Turkey (2003).

[PiZW03] G. Piccinelli, C. Zirpins and W. Lamersdorf: The FRESCO Framework: An Overview. Proceedings 2003 Symposium on Applications and the Internet Workshops (SAINT 2003 Workshops) Published by IEEE Computer Society, S. 120-123

[SABK98] R. Schmidt, U. Assmann, P. Biegler, R. Kramer, P. C. Lockemann, C. Rolker: The Interrelatedness of Component-oriented Systems And Work-flow-Management: What they can do for each other. Workshop on Com-positional Software Architectures. Monterey, USA, 1998, http://www.objs.com/workshops/ws9801/papers/paper099.doc

[SBMW96] W. Schulze, M. Böhm, K. Meyer-Wegener: Services of Workflow Objects and Workflow Meta-Objects in OMG-compliant Environments, OOPSLA 96, Workshop on Business Object Design and Implementation, San José, CA.

[Schm97] R. Schmidt: Component-based systems, composite applications and workflow-management. In Proceedings Workshop Foundations of Com-ponent-Based Systems, Zürich, Schweiz, 26. September, 1997, S. 206 — 214.

[Schu99] W. Schulze: Ein Workflow-Management-Dienst für ein verteiltes Objekt-verwaltungssystem. Dissertation, TU Dresden 1999.

[SeDH02] J. Seeley Brown, S. Durchslag, J. Hagel III: Loosening up: How process networks unlock the power of specizalization. The McKinsey Quarterly 2002 Special Edition: Risk and Resilence.

[W3C] www.w3c.org

[W3WS] http://www.w3.org/2002/ws/

[WeKl02] I. Wetzel, R. Klischewki: Serviceflow Beyond Workflow? Concepts and Architectures for Supporting Inter-Organizational Service Processes. 14th CAiSE. Springer Lecture Notes in Computer Science, Berlin, pp. 500-515, 2002

[WSDL] http://www.w3c.org/wsdl

[WSFL] F. Leyman, 'Web Services Flow Language', IBM Software Group specification, Mai 2001

[XML] http://www.w3c.org/xml

[YaPH02] J. Yang, M. P. Papazoglou, W. J. v. Heuvel: Tackling the Changes of Service Composition in E-Marketplaces, Proceedings of the 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce / E-Business Systems (RIDE-2EC 2002), San Jose, USA, 2002