Flexible Support of Inter-Organizational Business Processes Using Web Services

Rainer Schmidt
Department of Computer Science
University of Applied Sciences
Beethovenstraße 1
73430 Aalen
Rainer.Schmidt@fh-aalen.de

Abstract. Inter-organizational business processes which cross multiple organizations need new kinds of flexibility which are not properly supported by existing enactment architectures. Therefore, after analyzing these requirements, a new enactment architecture based on web services is developed. It uses a homogeneous and dynamic composition of so called aspect-element-oriented web services to support inter-organizational business processes.

1. Introduction

Today, enterprises must be able to quickly establish partnerships with other enterprises to combine their competencies. The set of enterprises participating in the business processes may change rapidly: there is a high fluctuation of partners. This permanent building and dismantling of partnerships is the key feature of inter-organizational business processes and necessitate new kinds of flexibility. To create an enactment architecture which supports inter-organizational business processes and their new flexibility requirements, this paper proceeds as follows. In section 2, new kinds of flexibility necessary for the enactment of inter-organizational business processes are analyzed; in section 3 further requirements for the enactment of inter-organizational business processes are identified. In Section 4, the enactment architecture, the so-called aspect-element-oriented schema representation is designed. It is implemented in Section 5 as a so-called composite application, using web services. Section 6 is dedicated to the evaluation of related research.

2. New kinds of flexibility in inter-organizational business processes

New kinds of flexibility are necessary to support inter-organizational business processes, which either augment or extend the flexibility needed for the support of intra-organizational business processes.
• Business process evolution
The enactment of inter-organizational business processes is subject to more changes than the one of intra-organizational business processes. There are more sources for change requests compared to an isolated environment. But more important, the changes differ in extent.

Changes of the business process model, the so-called business process evolution, imply the introduction of new modeling elements or rules. They may also imply the integration of new services.

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 contains an abstract view of the business process. It defines allowed paths for the execution of the business process. The schema 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.

• Service extensibility and integration
Above, business process evolution has been identified as an important requirement. Business process evolution implies that the set of services necessary to execute activities has to be extensible: New services have to be integrated due to changes to the activities. Therefore, an architecture for the enactment of inter-organizational business processes 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.

• Asynchronous service evolution
When enacting business processes across organizations, the underlying services may evolve asynchronously. There is no orchestrated evolution of services, because there is no central authority which could control evolution.

3. Further requirements for the enactment of inter-organizational business processes

• Enactment autonomy is defined as the capability to autonomously enact business processes. As participating organizations of inter-organizational business processes change quickly, all participants must be able to leave the partnership without
loosing the capability to enact their business processes and they must be able to integrate quickly into a new partnership.

- Knowledge encapsulation is necessary to protect the knowledge about business processes tailored to business needs, which is a decisive competitive factor. Due to the short lifetime of partnerships, the knowledge about business process design has to be kept secret from partners, which may soon participate in competing partnerships.
- In order to react to market changes it must be possible to quickly fulfill an increased demand of products. Therefore an enactment architecture for inter-organizational business processes must be highly scalable.
- Service autarchy is necessary to cope with business partner fluctuation typical for the enactment of inter-organizational business processes. Only if it is possible to quickly change the provider of a service, a quick adaptation to changing partners is possible.

4. The Aspect-element-oriented schema representation

Two basic architectures for business process enactment can be identified as shown in Fig. 1. They are called the direct and the indirect enactment architecture and differ in the amount of flexibility support.

In the direct enactment architecture business processes are enacted by a mechanism interpreting the business process schemata. Changes to business process schemas or instances can be implemented immediately because the schema is interpreted. On the other side, model evolution is not supported, because the interpretation engine has to be changed. The interpretation impedes service extensibility and integration, as the centralized enactment mechanism is not able to integrate new and formerly unknown services. Furthermore a centralized interpretation mechanisms is not able to cope with asynchronous, that means decentralized service evolution.
The scalability of the direct enactment architecture is rather limited as the interpretation is centralized and cannot be distributed. Because every step in the enactment of business processes requires the involvement of the enactment mechanism, a central bottleneck is created. Furthermore, the existence of a centralized enactment mechanism destroys the enactment autonomy and knowledge encapsulation as business process definitions are centralized and therefore not under control of the business process owner. Service autarchy is lost too, because the centralized enactment mechanism integrates all services used during business process enactment.

The other basic architecture is called indirect enactment. Here, enactment is done in two steps. In the first step, a so called schema representation is created. In the second step, the schema representation enacts the business process. Thus the enactment of business processes can be separated from the creation of the schema representation. The indirect enactment architecture offers only a limited flexibility for schema changes, because, the schema representation is monolithic and static. Service extensibility and integration are impeded because the schema representation is monolithic and the integration capabilities limited, the same applies for the asynchronous service evolution. Enactment autonomy and knowledge encapsulation can be achieved at least by workarounds such as partitioning the schema representation.

The result of our investigations shows, that neither the direct nor the indirect enactment architecture fulfils all requirements for the enactment of inter-organizational business processes. Therefore two different potential strategies exist. One strategy is to improve of the direct enactment architecture. However, there are requirements that cannot be met by this architecture such as enactment autonomy and knowledge encapsulation. The other strategy is to improve the schema representation of the indirect enactment architecture in order to support business process evolution. To do so, the former monolithic schema representation has to be split up into schema elements with an appropriate granularity.

First ideas concerning an appropriate service granularity for business process enactment can be found in [AWBB93] and [LeRo97]. An analysis why object-oriented software systems in general may become inflexible is the basis of Aspect-Oriented-Programming (AOP) [Kicz96]: Therefore, when designing a schema representation, the aspects of the business process have to be identified and separated. Five basic aspects of business processes are identified in [JaBu96]: the functional, control, informational, organisational and operational aspect. These “core-aspects” are orthogonal dimensions of business processes. An appropriate granularity can be achieved using aspect elements. 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.

Furthermore, it has to be decided whether a homogeneous or heterogeneous composition should be chosen. A heterogeneous composition means that one group of aspect elements plays an emphasized role. 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
Flexible Support of Inter-O rganizational Business Processes Using Web Services

5. Implementing the aspect element oriented schema representation by composite applications

It can be easily seen, that 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.

Composite applications using web services break with the thinking that applications and executables have to be one. There is no “executable” any more which contains the application functionality. A composite application is created by a set of interconnected and parameterized web services as shown in Fig. 3

Web Services [W3WS] 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]. The use of web services as implementation base helps to fulfill some of the requirements for the enactment of inter-organizational business processes identified above. Service
autarchy is fulfilled by web services, because they provide universal access to services. They also fulfill 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 extend the set of available services. Different versions of a web service can be differentiated by different name spaces. Therefore web services fulfill also the requirement of asynchronous evolution of services.

The adaptation of web services to an individual composite application 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 parameterization and connection information. The parameterization 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.

An example is given in Fig 3. There are two composite applications A and B. They are created using three web services s1, s2 and s3.

Fig. 3. Composite application

Composite application A is created by using web services s1 and s2 with the specialization information denoted with the context identifier ia1. The identifier is composed from a global and a local context identifier. The global context identifier ia associates the specialization information with the composite application A. The local context identifier denoted “1” determines the context within composite application A. The specialization of 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 ib. 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. 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.

The execution of aspect element oriented composite applications is done by running iteratively through 4 phases. 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. The first phase is the initiation phase: The service is started using the information supplied from the calling service. In the second phase, the specialization phase, the service is tailored to the individual needs of the composite application. In the third phase, the execution phase, the service is executed using the parameterisation and connection information, which is contained in the specialization information. 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.

6. Related research

In [AALS00] Message Sequence Charts and Petri Nets are suggested for the specification and verification of inter-organizational business processes. The focus of this approach is to verify the consistence between inter-organizational business processes and message sequence charts. In [MPSK98] a dynamic business process 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 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 FRESCO project [PiZW03]. Both eFlow, DySCO and FRESCO, however, do not provide an enactment architecture. In [NaLS03] the use of rules and events for business process enactment is proposed. There are also coordination related extensions to web services 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].

7. Summary

The enactment of inter-organizational business processes is crucial for enterprises which have to quickly adapt or create partnerships with other enterprises to support open process networks and virtual enterprises. The fluctuation of business partners creates new and extended flexibility requirements to the enactment of business processes. The key to fulfill all of these requirements is to use an aspect-element oriented schema representation as part of an indirect enactment architecture. 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.

8. References


[W3C] www.w3c.org


[XML] http://www.w3.org/xml