Web Services Based Architectures to Support Dynamic Inter-organizational Business Processes

Rainer Schmidt

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

Abstract. Dynamic inter-organizational business processes are necessary to enable the flexible creation of partnerships in areas such as e-commerce and supply-chain-management. Although many information system architectures for the support of static inter-organizational business processes exist, such architectures are still not available for supporting dynamic inter-organizational business processes. In this paper the special requirements created by dynamic inter-organizational business processes will be analyzed and the contributions of existing approaches and web services evaluated. Based on the paradigm of the composite application, an architecture designed to support dynamic inter-organizational business processes has been developed and will be introduced.

Keywords: Dynamic Inter-organizational business process, composite application

1 Introduction

Successful enterprises must be able to cope with quickly changing market requirements and keep pace with the accelerated speed of economic change in areas such as e-commerce, e-procurement, supply-chain-management, etc. Enterprises, for example, must continually seek out new suppliers, which provide better products, or be willing to cooperate with new partners to develop more efficient supply chains. Each new partnership implies the creation or adaptation of business processes that cross organizational boundaries, and are best identified as "dynamic inter-organizational business processes". The dynamic inter-organizational business processes entail a permanent building and dismantling of partnerships, which is the key difference between them and (static) inter-organizational business processes, which have existed since the 1960s and have been based on technologies such as EDI [EDI]. The change from static to dynamic inter-organizational processes is further promoted by the appearance of technologies that allow for the dynamic coupling of information and application systems. The Internet, the World Wide Web and, ultimately, Web Services [W3C] provide transparent access to services in distributed and heterogeneous environments. Web services based languages such as BPEL4WS [BPEL] offer the specification of business processes using web services. However, an architecture

M. Jeckle and L.-J. Zhang (Eds.): ICWS-Europe 2003, LNCS 2853, pp. 123–136, 2003.

which describes how to create process instances of dynamic inter-organizational processes, and how to execute them, still does not exist.

There is a plenty of research which covers the support of inter-organizational processes (also named cross-organizational processes, cross-enterprise processes etc. see "3 Existing Approaches for the Support of Static Inter-organizational Business Processes" later in this text). However, these approaches refer to static inter-organizational processes and not to dynamic ones. In order to analyse the requirements of dynamic inter-organizational business processes and to propose an architecture to support them, this paper will proceed as follows: first, basic terms and concepts in the area of business processes will be defined. Then, the requirements for the support of dynamic inter-organizational business processes will be identified. Using these requirements, existing approaches will be examined to determine whether they can provide at least partial solutions. This analysis is the basis for the development of an architecture to support dynamic inter-organizational business processes.

2 Basic Concepts of Business Process Support

A business process is a bundle of activities, which requires one or several inputs and which creates value for the customer. [HaCh93]. A business process is formalized using a business process model, as illustrated in Fig. 1. The business process model defines a set of elements which represent the parts of the business process. It can be seen as a kind of language description containing words and rules about how to combine words to phrases. In most cases this 'language' consists of graphical elements and rules on how to connect them.

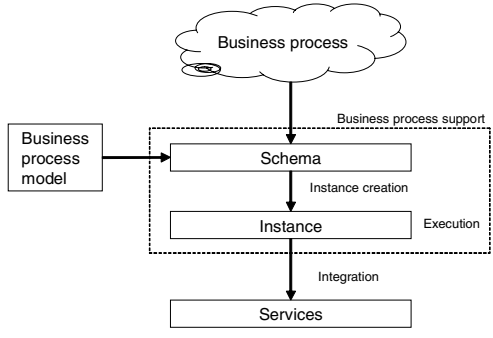

Fig. 1. Business process support

Using the elements and rules of the business process model, the real business process is formalized as business process schema (or schema for short). The business process schema abstracts the individual execution of the business process instance. Basically the schema defines the sequence of activities to be completed by the business process and the rules for carrying out certain activities in certain defined circumstances. For example a schema of an order processing defines the activities to be performed to process the order and rules i.e. "Reject the order if the ordered product is out of stock".

The business process schema is the 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. During execution of the business process instance, services are used to perform activities and to access data. The creation of instances and their execution is called, 'support of business processes'.

2.1 Requirements for the Support of Dynamic Inter-organizational Business Processes

Dynamic inter-organizational business processes are processes that are distributed over several organizations and show a high fluctuation of partners. The set of organisations participating in the definition and execution of business process is dynamic and not static. Organizations participate in the execution of a dynamic inter-organizational business process by executing one, or more, sub-processes and providing services for process execution even for sub-processes executed by other organizations. To clarify these points, we will introduce the scenario that is illustrated in Fig. 2. The scenario contains a mechanical engineering firm and an electronic control systems manufacturer that combine their core competencies. The mechanical engineering company profits from making their products more 'intelligent' with the integration of electronic control systems. The company for the production of electronic control systems gains access to new markets.

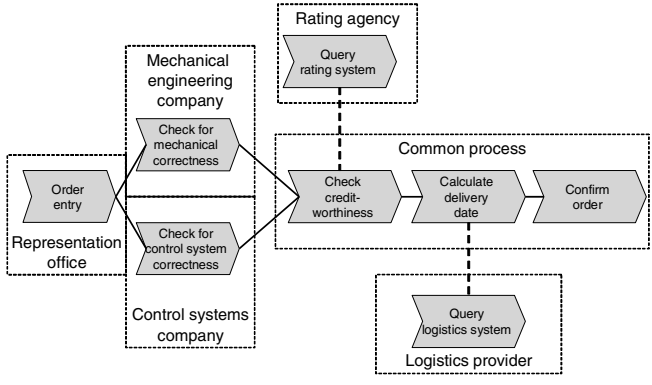

Fig. 2. Scenario for dynamic inter-organizational business processes

As both companies do not have infinite financial resources, they rely on representation offices to acquire orders. Only one of these representation offices is shown in Fig. 2. Acquired orders are sent to both companies to verify the technical details of the order from both mechanical engineering and the control systems perspectives. Then, an external rating agency is consulted in order to verify customer creditworthiness. This is done by querying the rating agencies rating system. The delivery date is calculated by querying the logistics system of an appropriate logistics provider. Lastly, an order confirmation is sent out (potential exceptions, e.g. a bad credit ranking are yet not covered). Six requirements for the support of dynamic inter-organizational business processes can be identified:

Process Autonomy
Process autonomy is defined as the capability of autonomously defining and executing business processes. Since the participating organizations of a dynamic inter-organizational business processes change quickly, all participants should be assured to retain their capability to define and execute business processes, even when they leave such a partnership. All participants must be able to enter into newer partnerships quickly. In the scenario, process autonomy would be destroyed by using a centralized business process support system installed at the mechanical engineering company. As process schemas and service descriptions are centralized at the mechanical engineering company, the control systems company loses the rapid response ability to switch to another partner.

Flexibility
Dynamic inter-organizational business processes are subject to more changes than static inter-organizational business processes since greater potential sources of change requests exist. Therefore, the support of dynamic inter-organizational business processes must be highly flexible.

Scalability
In order to react to market changes it must be possible to provide quick support for additional business processes or to augment the instances of an existing business process significantly. Therefore, scalability is an important requirement of supporting dynamic inter-organizational business processes.

Service Autarchy
Service autarchy is defined as the capability of independently associating the services needed to support business processes or sub-processes of them. Service autarchy guarantees that each organization participating in a certain business process can use the necessary services to execute their business processes without the involvement of a third party. Service autarchy is necessary to cope with business partner fluctuation typical to dynamic inter-organizational business processes. Only when it is possible to quickly change a service provider is it possible to change a business process quickly. In the scenario shown in Fig. 2, service autarchy would be lost by using a centralized Enterprise Application Integration (EAI) system at one company. For example, an EAI-system is installed at the mechanical engineering company. Services s1 and s2 provided by the rating agency and the logistics provider are centrally integrated. This means that they can not be directly accessed; every usage of the services would have to pass through the EAI system. This could be disastrous when the controls systems company switches to another partner because access to the services would be lost.

Service Extensibility and Integration
Because dynamic inter-organizational business processes are subject to continual change, the set of services necessary to enable them is highly dynamic as well. New services must be continually integrated due to changes in the business processes. Therefore, an architecture to support inter-organizational business processes must be capable of integrating new and previously unknown services. The services may be highly-distributed and heterogeneous, because different organizations use different infrastructures. Service extensibility and integration describes the possibility of

quickly and transparently switching from a service, referred to as s2, and provided by the old logistics, to the service s3 provided by the new logistics service. Service s3 may be implemented in a totally different way than s2.

Asynchronous Service Eevolution
When dealing with dynamic inter-organizational business processes, services evolve independently. There is no centralized institution coordinating the evolution of services. In the scenario, illustrated in Fig. 2, the rating agency could introduce new and enhanced versions of its rating service s1 called s1'. This should not influence the established business process in any way. Asynchronous service evolution is defined as the capability to cope with services which evolve independently and without central coordination.

The requirements for supporting dynamic inter-organizational business processes can be divided into two groups. Process autonomy, flexibility and scalability are process oriented requirements. They deal with the definition, management and execution of business processes. The second group are service oriented requirements, which include service autarchy, service extensibility and integration and asynchronous service evolution.

3 Existing Approaches for the Support of Static Inter-organizational Business Processes

The support of (static) inter-organizational business processes is a frequently mentioned topic with many concepts. To identify possible contributions to the support of dynamic inter-organizational business processes, a selection of approaches will be examined using previously identified requirements.

The first group of approaches tries to create a support for inter-organizational business processes based on CORBA [OMG] such as [VoWe99]. The most advanced approach is the WorCos-concept [Schu99]. The primary goal of the WorCos-concept is the support of business processes on the basis of a workflow service, which is integrated into the Object Management Architecture (OMA) [OMG]. In [SBMW96] and [Böhm97], a proposal for a CORBA-Facility [OMG] is made. WorCos is a purely object-oriented approach. Business processes, are represented as CORBA objects, created by compilation. The objects contain nearly the complete functionality to execute a specific business process, such as role associations, process state and meta data. Therefore, the execution of business process instances is centralized in these objects. The Mentor project [WoWe97], [WeMW98] is aimed at supporting business processes organization-wide. The fundamental idea of the 'Mentor Project' is to model business processes in the form of state and activity charts [Hare88]. In this manner, different schemas can be easily distributed and executed. However, the execution of the business process instances is centrally managed. The Meteor- and Meteor2-Projekt [Wang95] creates a fully distributed process support for company wide business processes based on CORBA. Integration into the WWW is described in [DKMS96], [MPSK98]. In Meteor, business processes are represented in a Workflow Intermediate Language (WIL). This specification is used to generate task managers

which control parts of a business process. Task managers are independent execution units, which may be distributed throughout an organization.

When the approaches listed above are carefully analyzed, two basic architectures can be identified. They are called, direct and indirect instance creation. In the direct instance creation architecture process instances are created directly from the process schema, and no intermediate structure exists. The process instances are created by interpreting the process schema as shown in Fig. 3. The engine which interprets the process schema also integrates services, e.g. s1 and 2. This architecture can be found in the Mentor approach.

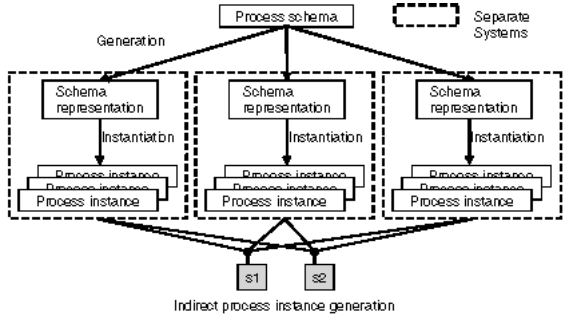

Fig. 3. Direct and indirect process instance creation

High flexibility is the primary advantage of the direct instance creation. Changes to the process schema can be immediately implemented because the schema is interpreted. However, this advantage is acquired by accepting the disadvantage of limited scalability since the interpretation is centralized and cannot be distributed. Since every step in the execution of the business process requires the involvement of the interpreter a central bottleneck is created. Furthermore, the existence of a centralized engine obliterates the process autonomy and service autarchy. Changes to sub processes can only be implemented by interacting with the centralized interpretation engine, which also integrates all the services used during process execution. The interpretation also impedes service extensibility and integration, since the interpretation engine is not designed in an extensible way. The same applies for asynchronous service evolution.

The other basic architecture - indirect instance creation - is also illustrated in Fig. 3. Here, process instances are created in two steps. In the first step, a so-called schema representation is generated. It is a template used for the automatic generation of process instances called instantiation. In the second step, the schema representation is used to create business process instances, which can be executed independently. Both instantiation and execution can be separated from the generation of the schema representation. Furthermore, services such as s1 and 2 have, for example, have not to be centrally integrated.

Indirect instance creation can be found in the Meteor and WorCos approaches. In WorCos, the schema representation consists of an object class created from the process schema. The instance creation is done in WorCos by creating instances of the object class. The use of two steps to create process instances is the reason for the high scalability of the indirect instance creation. The creation of a schema representation

can be separated from instance creation and execution. By copying the schema representation and distributing it, process instances can be executed on a multitude of systems. Thus a high scalability is achieved because many process instances are executed in parallel on separate systems. However, indirect instance creation offers only limited flexibility, as changes to the process schema cannot be implemented incrementally. Process autonomy can be achieved by creating separate (sub-) schema representations for each sub process. In this way process autonomy can be achieved. Service extensibility and integration are impeded by the generating mechanism for the schema representation. The set of services is fixed, with 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 instance creation. By comparing the direct and indirect creation of process instances it can be easily seen, that there is a dilemma between flexibility and scalability. Direct approaches provide flexibility but limited scalability; indirect approaches provide scalability but only limited flexibility.

3.1 Web Services

Web Services [W3WS], [GrSi02] provide transparent access to services in a highly distributed, heterogeneous and dynamic environment. This achievement has been a goal within the field of computer science research for a long time. Web services realize the concept of a component as software unit, which is transparently accessible and makes explicit all context dependencies [Szyp96], [ScCi96]. An essential ingredient for the success of web services is the so-called, loose coupling of services. The implementation of loosely coupled services can be exchanged at run-time without side effects for the web service user. Loose coupling is crucial for service autarchy, because such services can be quickly integrated or sorted out. Furthermore, the name space concept already introduced with XML [XML] is very useful for coping with asynchronous service evolution by creating a decentralized versioning mechanism. Different versions of a web service are differentiated by different name spaces. The basic Web Services technologies provide no direct business process support. When they are combined with the architectures identified so far, the direct and indirect instance creation, only an improved service integration and support for service evolution is achieved. Therefore, applying Web Services to those architectures is not a remedy for the fundamental deficiencies found in these approaches.

3.2 Web Services Based Approaches for Business Process Support

Soon after the definition of basic Web Services technologies such as SOAP [W3C] it became more and more obvious that web services also have to provide concepts for orchestrating and coordinating web services in order to support business processes.

The Business Process Execution Language for Web Services (BPEL4WS) [BPEL] is based on WSFL and XLANG. It allows the specification of business processes and the definition of web services to be used in executing a specified business process [LeRo02].

BPEL4WS, WSFL and XLANG are language definitions which provide a set of elements and rules to describe business processes. Therefore, they can be thought of as a type of business process model. However, none of them provides concepts for executing and instantiating the defined schemas.

3.3 Execution Environments

A first approach for supporting the execution of business processes specified with the BPEL4WS is the IBM Business Process Execution Language for Web Services, JavaTM Run Time (BPWS4J) [BPWS4J]. The BPWS4J executes business processes using 3 documents. The process schema is represented in a BPEL4WS document. A WSDL-document [WSDL] specifies the interface that the business process provides 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. Thus, BPWS4J is an approach using the direct instance creation. Therefore, the advantages and disadvantages of the direct instance creation approach also apply to BPWS4J. It is flexible, because process changes can be implemented immediately. However this flexibility is acquired in exchange for limited scalability.

3.4 Summary

The result of our investigations can be summarized as shown below in Fig. 4. Direct instance creation provides flexibility but fails to fulfil all other requirements. Indirect instance creation offers scalability and process autonomy, but does not provide service oriented requirements. Web services are not designed to support business processes directly. Therefore, it is not surprising that they fulfil only the service oriented requirements.

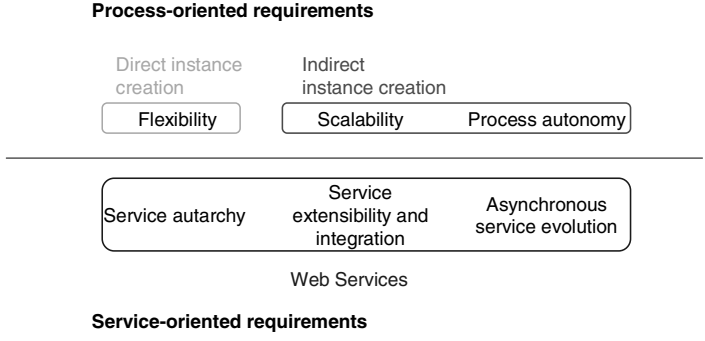

Fig. 4. Fulfilment of the requirements for dynamic inter-organizational processes

To create support for dynamic inter-organizational business processes by fulfilling all requirements, one has to combine the advantages of direct and indirect instance creation and combine them with the benefits gained from employing web services.

4 Support for Dynamic Inter-organizational Business Processes by Using Composite Applications

To create an architecture which escapes from the dilemma of trading off either scalability or flexibility and to fully make use of the advantages offered by web services, two different strategies are possible.

One strategy is to improve the scalability of direct instance creation architectures. For example, an attempt could be made to use several interpretation engines instead of only one. However, this does provide an answer to the fundamental problem that the 'flow of control' of the business process always returns to the interpretation engine. Therefore the centralized engine remains the bottleneck.

The other alternative is to use the indirect instance creation architecture and to make the schema representation incrementally adaptable, so that it can be quickly modified according to a changed business process. To achieve this, the paradigm for creating a schema representation has to be changed. Up until now, all the approaches resulted in the creation of a schema representation as a classical monolithic application. One has to go through a kind of edit-compile-link cycle to change the schema representation.

A paradigm which avoids these limitations is the composite application [Schm97]. Composite applications break with the thinking that applications and executables have to be one. A composite application is created by a set of interconnected and specialized services as shown in Fig. 5.

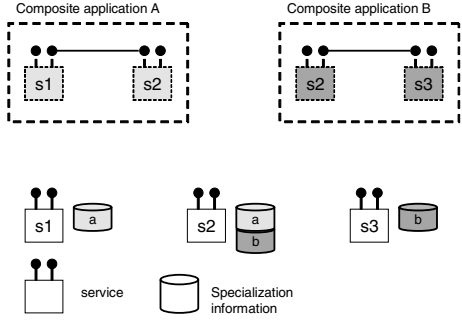

Fig. 5. Composite application

There is no central "executable" anymore which contains application functionality. All functionality remains in services which can participate in different composite applications. The services are integrated into composite applications via specialization. The specialization is done by applying specialization information to the services. The specialization information contains both parameterisation and connection information. Parameterisation information adapts the service to the individual needs of the composite application. The connection information contains information about the connection of the service with other services within the context of the composite application. The specialization information is not centrally stored but attached to the services. Therefore, every service "knows" what to do. An example is given also in Fig. 5. There are two composite applications A and B. They

are created using three services s1, s2 and s3. Composite application A is created by using services s1 and s2 with the specialization information denoted as “a”. The specialization information “a” of s1 contains the connection information representing the connection between s1 and s2. Furthermore, both services are parameterized. Composite application B is created by using services s2 and s3 using specialization information denoted with a “b”. The paradigm of the composite application is used to create an incrementally adaptable and thus flexible schema representation. The starting point for this is the decomposition of processes and subprocesses into so-called representation elements. Representation elements can be easily identified by separating the business process functionality according into so-called aspect-elements [ScAs98]. Aspect elements are not further dividable; they are atomic parts of business processes which contain only functionality of one so called aspect [JaBu96]. Aspects are disjoint sets of elements of business process or workflow models. There are 5 basic aspects: the functional, control, informational, organisational and operational aspect [JaBu96]. The functional aspect describes how a business process is composed of sub-processes. 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.

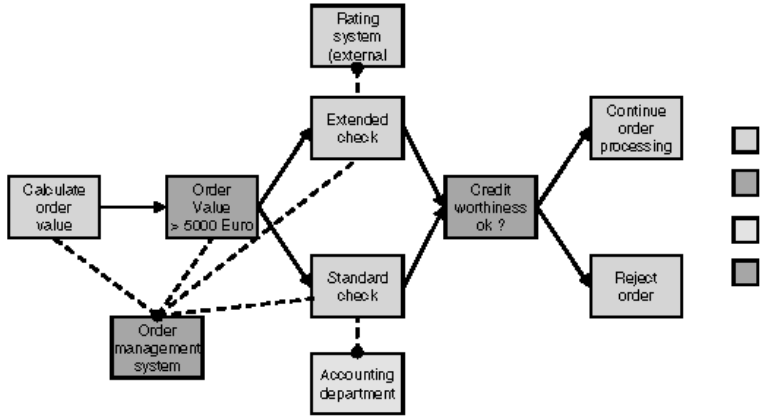

Fig. 6. Decomposition of business processes into representation elements

In order to illustrate the generation of a schema representation, a small fragment of a business process is transformed. The fragment defines, that above an 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. Two representation elements connected in this way are meant to be executed consecutively. Dotted lines indicate client-server relationships.

The first step to creating a composite application is the generation of representation elements from a business process schema provided as BPEL4WS document for ex-

ample, as illustrated in Fig. 7. The second step for creating a composite application is to search for web services which can implement the representation elements. This may be done by querying a service repository such as UDDI [UDDI]. For each representation element, an appropriate service has to be found. In the scenario below, the service associated with the representation element is delineated below the representation element. In most cases, there is not a perfect fit; therefore, the search has to also include also services which can be properly adapted using specialization. Furthermore one service may fit well with several representation elements.

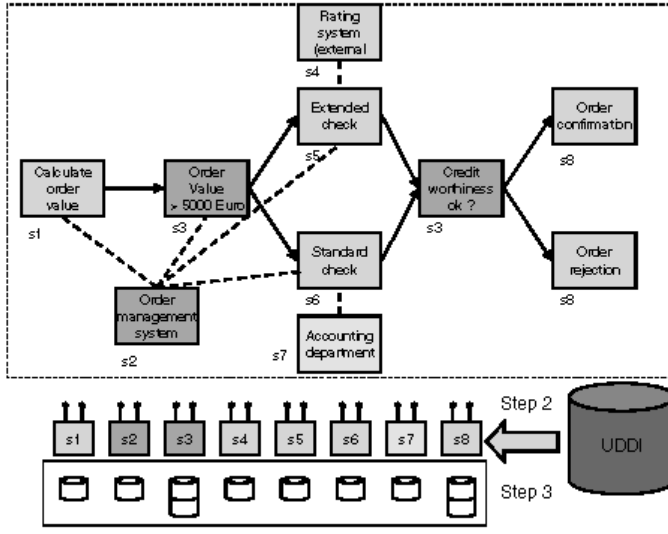

Fig. 7. Generation of a composite application

For example service s3 can be used to implement a check of whether the order value is greater 5000 euro and a check of whether a customer is credit-worthy. This is possible, because the service may be adapted via parameterization to the respective criterion. The first use of the service tests the criterion order value, the second checks the credit-worthiness. The same applies to service s8 which is used to implement both the order confirmation and potential order rejection.

In the third step, the specialization information is generated, as shown in Fig. 7. Appropriate parameterization information has to be provided to adjust the service to the individual needs of the business process. For example the order value which should initiate a detailed investigation of the customer's credit-worthiness can be adjusted. At the same time, connection information also has to be generated according to the connections of the representation elements in the business process. It is important to notice, that services s3 and s3 have 2 sets of specialization information because both are used twice. The types of use are differentiated using context identifications, which are issued by the antecedent service. For example s1 issues context identification 1, s5 and s6 issue context identification 2. In this manner, s3 will test the order value if it is called by s1 and test credit-worthiness if it is called by s5 or s6.

5 Conclusion

Dynamic inter-organizational business processes are crucial for enterprises which have to quickly adapt or create partnerships with other enterprises: Partners are replaced in existing partnerships and new partnerships are constantly being built. This fluctuation of business partners creates new or extended requirements. Two groups of requirements have been identified. Process oriented requirements concern the definition, management and execution of business processes: process autonomy, flexibility and scalability. The second group are service oriented requirements: Service autarchy, service extensibility, integration, and service evolution. The analysis of existing approaches has shown that neither direct, nor indirect instance creation can satisfy all requirements or preserve the benefits offered by web services. The direct instance creation architecture offers flexibility, but lacks scalability and process autonomy; the indirect instance creation architecture offers scalability and process autonomy, but lacks flexibility. The key to escaping this dilemma is to use a flexible and adaptable schema representation based on composite applications. In this way, it is possible to create an execution architecture for dynamic inter-organizational business processes, which is both flexible and scaleable. Process autonomy is fully achieved, as the composite application can be fragmented on a representation element level basis. The use of representation elements as a paradigm also assures that the process autarchy provided by web services is maintained. The same applies for service extensibility and integration. In addition the ability of web services for supporting the asynchronous evolution of services is also preserved.

The next steps will comprise the introduction of tracking and tracing mechanisms and the integration of legacy systems. Tracking and tracing mechanisms provide the capability to track the execution of business process instances and to create a log of the business process execution which is necessary in many business areas. The integration of existing applications systems poses the challenge that they contain functionality which does not separate aspects and even aspect elements. Therefore a system of wrappers and mediators has to be developed, which allows the integration of legacy systems.

References

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

[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 Modeling Language (BPML).

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

[Böhm97] M. Böhm. Objektorientierte Implementierungstechniken für Workflow-Management-Systeme in OMA-konformen Architekturen. In [JaBS97].

[Chap96] D. Chappell: Understanding ActiveX and OLE. Microsoft Press, Redmond, 1996.

[CiSc96] O. Ciupke, R. Schmidt: Components As Context-Independent Units of Software. WCOP 96, Linz 1996. Special Issues in Object-Oriented Programming. Workshop Reader of the 10th European Conference on Object-Oriented Programming ECOOP96. Dpunkt.verlag, Verlag 1996.

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

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

[HaCh93] M. Hammer, J. Champy: Reengineering the Cooperation, A Manifesto for Business Revolution. Harper Business, New York, 1993.

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

[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), December 1996.

[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

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

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

[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.

[ScAs98] R. Schmidt, U. Assmann. Extending Aspect-Oriented Programming in Order to Flexibly Support Workflows. In Proceedings of the ICSE98 AOP Workshop, Kyoto, Japan, April 1998, S. 41 – 46

[Szyp96] Proceedings and Summary of ECOOP'96 Workshop: Component-oriented Programming (WCOP 96), 10th European Conference on Object-Oriented Programming, Linz, Austria, July 1996.

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

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

[Snel02] J. Snell: Automating business processes and transactions in Web services ftp://www6.software.ibm.com/software/developer/library/ws-autobp.pdf

[Szyp98] C. Szyperski: Component Programming, Beyond Object-Oriented Programming. Addison Wesley, New York, 1998.

[W3C] www.w3c.org

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

[Wang95] X. Wang: Implementation and Evaluation of CORBA-Based Centralized Workflow Schedulers. Master's thesis. University of Georgia, August 1995.

[WeMW98] J. Weissenfels, P. Muth, G. Weikum: Flexible Worklist Management in a Light-Weight Workflow Management System. In Proceedings of EDBT Workshop on Workflow Management Systems, Valencia, 1998.

[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

[WfMC] Workflow Management Coalition: http://www.aiai.ed.ac.uk/project/wfmc/

[Wodt96] D. Wodtke: Modellbildung und Architektur von verteilten Workflow-Management-Systemen. DISDBIS 31, infix Verlag, Sankt Augustin, 1996.

[WoWe97] D. Wodtke, G. Weikum: A Formal Foundation for Distributed Workflow Execution Based on State Charts. In Proceedings ICDT 1997, S. 230 – 246.
[WSDL] www.w3c.org/wsdl
[WSFL01] F. Leyman, 'Web Services Flow Language', IBM Software Group specification, Mai 2001
[WWWK96] D. Wodtke, J. Weissenfels, G. Weikum, A. Kotz-Dittrich: The Mentor project: Steps towards organizational-wide workflow management. In Proceed-ings 12th IEEE International Conference on Data Engineering, 1996.
[XML] www.xml.org