Orchestration Definition from Business Specification

Charif Mahmoudi
Logics, Algorithm, Complexity Laboratory
LACL, Paris 12 University
Creteil, France
charif.mahmoudi@lacl.fr

Fabrice Mourlin
Logics, Algorithm, Complexity Laboratory
LACL, Paris 12 University
Creteil, France
fabrice.mourlin@wanadoo.fr

Abstract—The implementation of service orchestration is often seen as a convoluted process for business analysts, lead developers and architects. In this document, we propose a new approach based on a continuous process starting from the analysis phase to the architecture phase as an attempt to standardize the implementation of service orchestration. Our ultimate goal is to have a BPEL (Business Process Execution Language) script which will be interpreted by an engine residing inside a middleware generating a composition of elements where each element can be considered as an independent component equipped with a Web service. Orchestration definition contains several facets such as logical, pragmatic and architectural aspects; each of them is complementary and the interaction between them usually raises conflicts. In our approach, these issues are addressed and solved by adaptation rules and the problem of adapting the software architecture onto a physical architecture is solved by the pragmatism method.

Keywords-SOA; Architecture; Web service orchestration; business process design and specification

I. INTRODUCTION

The component-oriented approach has emerged and has become widespread in the industry to meet the scalability of information systems [1]. It reduces software costs and allows rapid adaptation to changing business and technological developments. It also enables software components to create highly modular and integrated. The development of certain parts of the information system may be too independent.

This component-oriented approach allows governing the evolution of technical and functional information system based on standard software. In addition, it covers all aspects of development and life cycle of the software. With the emergence of the component-based modeling paradigm, the OMG (Object Management Group) did not remain inactive and has proposed a new architecture based on MDA rules [2]. The development has facilitated UML modeling components. In 2001, the OMG defined the MDA approach with the aim to facilitate the integration of applications and make the specification of independent application development technologies. It also sets rules for mapping the standard specifications of different technologies [3].

UML Modeling tools generate the code source structure of applications. There is a transformation of a logic model to design model to the platform on the basis of design pattern templates and code [4].

The SOA is not far away removed from the component-oriented approach; see Peter Herzum [5]. He is one of the first authors having clearly defined the concept of components and component architecture. From his point of view, there are three types of components: Components "business process", components "business entity" that implement a core business concept, and finally, the component "business tool" used in various system components. He proposed to build a system specification based on four models. A business process model is used to identify components known as "business process" that manage one or more use cases. A model of "business entities" supports one or more business processes. A model is created to define business events. Another model is created for the definition of business rules.

The component-oriented approach has been developed within companies, but the purpose of sharing common components is often wishful thinking. Projects are organized into business lines. The application needs vary greatly over time. The services are requested too often, and the code of common components is duplicated and modified directly in new applications as alternatives. Reuse requires the establishment of specific resources such as the development of cataloging tools, dissemination of information about the components, creating a team to administer the transverse components. It also requires the definition of a target upstream of urbanization of the information system. We present in this contribution our approach to defining orchestration from business specification, and to mix it with other reused components. In the next section, we explain our design process for SOA architecture. Then, we give details about the semantic model of our approach and pragmatic model also. The following section is about logical model and how we declare it. The last part is about architecture and implementation of orchestration. Finally, we provide a case study of our approach.

II. DESIGN PROCESS FOR SOA ARCHITECTURE

The concept of enterprise architecture management was gradually adopted in enterprises to address the problems of organization and urban information system. Different methodological approaches have been developed or

 ISBN: 978-1-61208-230-1

framework to build and maintain this architecture such as Zachman [6], [7] and TOGAF [8], [9] or EUP (Enterprise Unified Process) [10], [11]. Our approach provides a method for developing SOA and managing the complexity of the enterprise by integrating the evolution of new technologies. It is based on modeling different aspects of the system and a process of construction and derivation of the models. Basically, there are two views. An external view for describing the company level: business data manipulation, organization and business processes. An internal view can gradually develop the system: logic model and technique to build the software deployed on the hardware. The physical model describes the implementation and deployment. It extends to the logical architecture with the definition of service components and the technical architecture design. Our approach is providing a comprehensive urbanization of information systems by using a semantic model and a pragmatic model. We add to this a logical architecture with a design of the technical architecture.

A. Several models for a given project

We start our approach by a first pre-model, which is used to define a common vision to the various players in the enterprise. The description languages used are UML [12] and Business Process Modeling Notation (BPMN) [13]. The pre-model provides a dictionary of terms in the application field, an analysis of objectives and business needs. We identify high-level processes and key use cases and the fundamental business rules.

Our semantic model is intended to describe the basic business concepts of the company. This model can be established at two levels: The overall level contains the definition phase of urbanization. The local business domain contains the definition of business service. The purpose of the construction of these models is to achieve stability of business concepts. We build mostly by diagrams such as UML class diagram semantics, OCL constraints. For instance, Figure 1 provides a semantic model about bank operations:

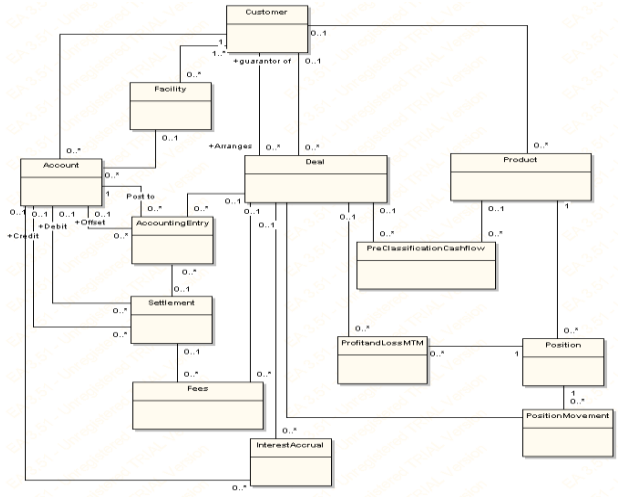

Figure 1. External view - business class diagram as semantics model.

Behind this kind of diagram, vocabulary is bound and a first set of constraints is taken into account. This business model is the knowledge of the company. This business model acts as a common language between all company projects. It can be built by successive iterations. In that diagram, Semantic classes and main attributes are defined. The life cycle of the business classes is also described. Relations between business classes are also provided clearly. Relations could be evolved with precise detailed information. For instance association between Deal class and PreClassificationCashflow class can change into a composition under business conditions.

The different projects feed into the common semantic repository. The difficulty of this analysis involves the construction of an observation with no prerequisites. The aim is to describe the business concept and not handled the technical which has been used in existing projects. The management of business objects needs a workflow description. Also, we have added such diagram attached to the fundamental business class. As an example, we give a description of the Order Fulfillment Process of a bank product (OFP) (Figure 2). The diagram shows that secondary business classes can be added for describing the process. It also focuses on the responsibilities of each step of the workflow.

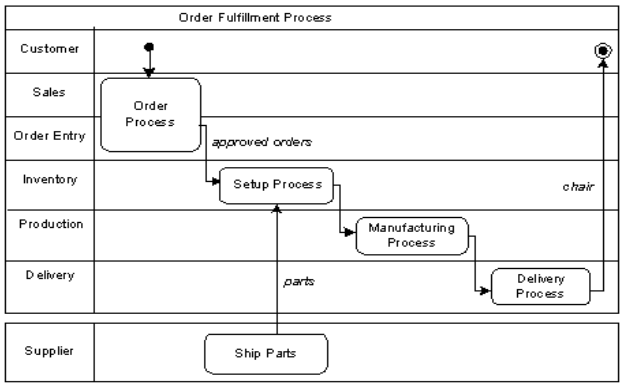

Figure 2. Workflow with partitions showing who does what work.

This global diagram provides main collaborations into business process and precisely causality between main events in the process. It also highlights synchronization between processes. This automaton is generally deterministic. Controls can be done with other workflows and conflicts are then detected which improve our models.

Business modeling is to improve the abstract concepts. The diagram should be simple and generic. It can anticipate the consideration of future developments.

The repository contains semantic early different business areas and key objects. It is enriched in with new projects. A review of models is to perform when they are stable.

 ISBN: 978-1-61208-230-1

B. Architecture and semantics

Then, the specification of services is driven by the business analysis. It involves business managers and technical managers. It requires defining relationships between the UML diagrams that were constructed. Then, all descriptions can be observed as a multi layers diagram as follows (Figure 3).

A business layer is based on the functional layer, this functional layer depend on the application layer. The implementation of this application layer is described by the technical layer.

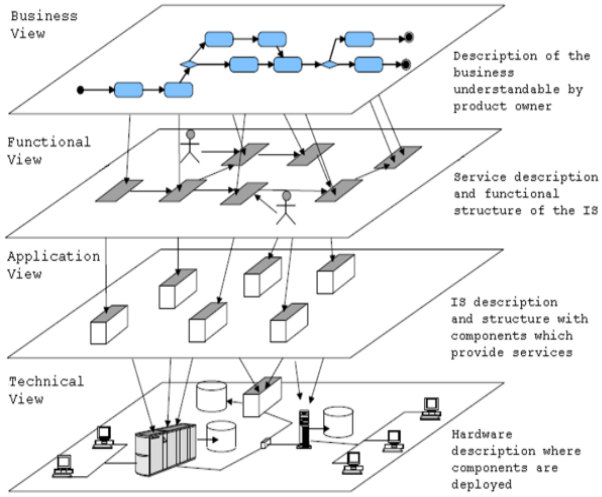

Figure 3. Methodology approach.

The use of four views allows traceability of the business to the computer. It provides the logical sequence between the business views, functional and application.

This top-down functional approach may have a downside: how to structure the architecture of services into a stable structure? And witch levels of abstraction in the information system will we consider. Our answer is a dual approach. First, learn the basic services starting from the semantic model (Figure 4, functional view) and also complement the services based on the principles of the organization (Figure 4, application view): use case of the information system and business process details.

The modeling principles presented apply with a global reach and local levels. Of course, there are analysis and design and the need to reorganize and streamline processes. Use case diagrams of the information system are the business functional requirements that must leave the system in a consistent state. It ensures the unity of actor, unity of place, time unit.

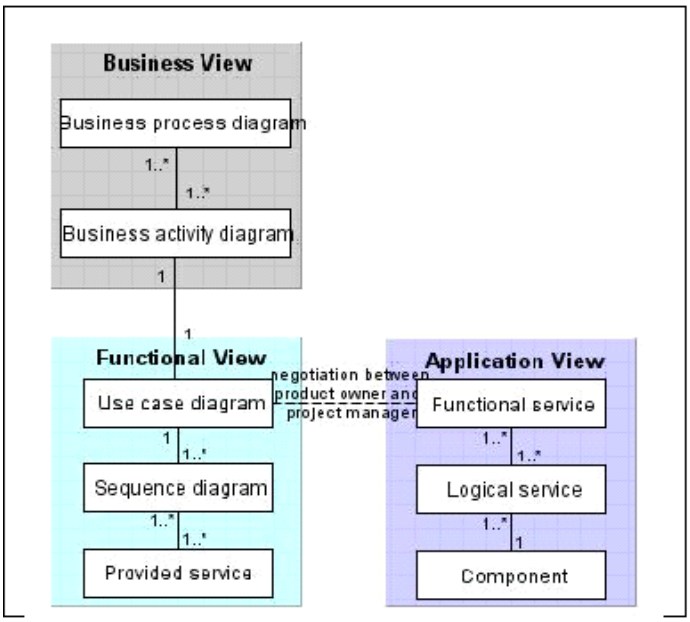

Figure 4. UML models used in our approach.

Our method for identifying use cases is classic. It identifies the actors, list the types of events. Finally, we deduce the interactions with the system. Then we structure all the use cases to a hierarchy or order planning for a future project.

Of course, in some projects, it is necessary to take into account organizational or operational constraints. In this case we show a view of the organization. It describes the process in relation to the business organization.

Other constraints must also be considered as the geographical dispersion of actors and business processes. These features may require significant optimizations. For example, the choice of appropriate new technologies can help streamline and simplify processes. For example, the nomadic operation belongs to that kind of constraints.

The underlying logical model is intended to specify and organize the services of our SOA. This is accomplished by the use of semantic diagrams (such as Figures 1 and 2). This defines the logical architecture that will be derived within the meaning of the Model Driven Architecture (MDA) approach [14] in software components. The word “logical” denotes the sense that the logic model should remain free of any technical choice. It contains a Platform-Independent Model (PIM) [15]. They can be derived to different technical platforms: J2EE, .NET, ESB, Web services, etc. Our logical architecture defines the components and services based on semantic aspects, pragmatic aspects and geographic aspects.

Our logic model is not deprived of any technical concern. It must produce a coherent model; this model must be implementable effectively while respecting technical choices.

C. Multi layer architecture

We structure our layered architecture. We distinguish a logical level, related to semantic classes, an organization

 ISBN: 978-1-61208-230-1

level, linked to the workflow or orchestration and finally a technical level related to the problems transverse. Our service concept is seen as elementary grain architecture. Services are provided at the logical level. They correspond to elementary operations directly related to the state machine of the main class of business concept. These operations are the transitions of this automaton. They normally have been specified in the semantic model.

The types of data exchanged between services must be precisely specified. They correspond to the design pattern DTO (Data Transfer Object) J2EE applications. It is the pivot language used in the process orchestrations. They reduce the number of parameters of services.

This data are formalized as classes belonging to a mapping utility. They are grouped in a factory to make them accessible to all services. These classes allow access to properties using getter and setter. All instances of those classes can be exposed through the use of context.

All components must provide at least one interface, which is to say all the public services exposed to the outside. The interfaces are not accessible. A component can provide different interfaces for different tasks of the components. A component also includes one or more data structures combining the exchange of trade data structures internal components. After deployment time these components will be exposed via a component server.

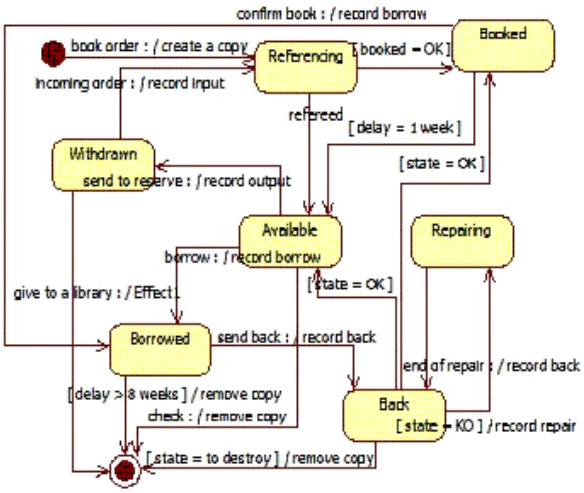

Figure 5. State chart of a copy

Use cases should have been structured to eliminate any redundancy. Each use case is then derived in an elementary workflow where each activity is a candidate to become distributed logical service. Basic activities correspond to simple exchanges with users. The others are often called business services.

III. SEMANTIC MODEL OF OUR APPROACH

Our semantic model is intended to gradually create a stable business model describing the general business concept business fundamentals. It corresponds to the concepts and business objects of the project field. It is described with UML notation: class diagrams with semantic attributes, relations between the business concepts, business rules that constrain them, the life cycle of the business classes.

A. Constraints on semantics model

The requests on the semantic model are: tractability upstream. It is useful to be able to justify the model in relation to its inputs (functional requirements, legislation, regulations, etc.). Other requirements relate to restitution: the diagrams must be interpreted in natural language. We must keep the synonyms of the terms in a thesaurus. Finally, the model should express the semantics of the domain while excluding of any other aspect.

Gradually, it evolves the model is documented in the project including different aspects such as the number of handled instances, their persistence...

The quality of such a model is assessed with reference to classical properties. The non-coupling expresses that each capture a single semantic entity. The homogeneity requires that we do not aggregate various aspects of business semantics. Sufficiency occurs when classes are all the information. Completeness is achieved when all the relevant features are included in the model.

This model is important because it is a communication medium between the project partners. Moreover, it must be easily usable by all members intervened in the project, whether internal or not.

Constraints and business rules are encapsulated in classes. The constraints are described as the sources of method or attribute.

The life cycle of business objects is described using finite state automata. Its purpose is to identify all the events changing the state of business objects and operations of a semantic nature. A second goal is to identify all the disturbances affecting the cycle of the object: the trigger events, operations performed during the transition.

B. Example of semantics model

We have studied workflow of copies of books which are managed into a library. The library has several sites into a town and books can be transferred from one site to another if there are not borrowed more than eight weeks. Other business rules are defined by business expert. This kind of diagram (Figure 5) is an ideal support for expressing rules and constraints because all existing cases are taken into account.

When constraints change, such diagram catches all new constraints, even if they involve the refactoring of the whole diagram. Because this diagram is linked to a business class, new methods enrich its behavior. Of course, such a description is rich in information and can update the business class diagram. These events are in addition to methods for the question of the life cycle of objects. Then we created a

 ISBN: 978-1-61208-230-1

package diagram by business domain. This constitutes around major business classes such as Library, User, Copy, etc. The semantic model is valid by reviewing models. It is also useful in cases of unit test on the model. This means checking state machines and the events received and forwarded. The model is simple and it is likely to be stable.

IV. PRAGMATIC MODEL AND PHYSICAL DISTRIBUTION

A. Process Analysis in UML

UML provides good modeling tools [16]. Use cases provide an overview of the system: reactions reports by the needs of actors. They can be used at two levels. Informally, they are used to identify the processes and use cases in a phase of reflection. Finally, more formally, they are useful for describing use cases of information system.

Activity diagrams provide a graphical representation to represent the largely consensual process. Sequence diagrams are used to describe the use case scenarios. But they are limited in their use to cases nominal and do not necessarily provide added value. The state diagram can also model the status of an activity of a business use case. They clear the events that affect its performance. Class diagrams are used to rank the players.

The difficulty of the analysis process is the level of granularity of modeling. The distinction is between processes and sub processes, tasks and activities. We often encounter a meta-level model of the company to agree on the level of detail of this modeling. The issue is the granularity of the service and its reusability often introduces unnecessary and lengthy discussions.

In this type of study, there are little methodological approaches completely formalized and really consensual. The existing methodological approaches are usually the owners of these processes. We preconize to use an open source approach to be able to use instrument allows fully shared processes by the tools.

B. The organizational view

1) Process definition

It contains descriptions of the process. This description comprises both system responses to external events and also the information flows (flows, object flows, internal events), it also includes coordination between the activities of different actors.

The view also contains organizational decisions and their explanations. This includes configuring services, distribution of responsibilities, the profile of players and possibly other constraints such regulations. This view is also about the definition of classes related to problems of organization and management. It also contains classes related to business events that are in the semantic model.

We consider a process as an ordered set of activities to produce a result: the production or transformation of an object. Our pragmatic model describes the process the processing steps acting on objects in the semantic model are already described as state machines associated with these objects.

Activity corresponds to an action or a set of actions. The mastery of activities requires organization and rules that are not present in the semantic model.

The process space is hierarchical, it is important to begin the descriptions of the most important processes. It is unnecessary to describe the process with a full level of detail rendering them incomprehensible except to experts. We establish a general map of the process. We determine the key process, that is to say those criticisms vs. strategic objectives. It is important to analyze the risks.

Our approach to analyze the process remains a classic. First, we outline the beginning and end of the process, and then describe the goal. This means knowing the customer's expectations. In addition, we describe the interface with other processes. We detail the resources used: objects manipulated. We add the traceability rules. Finally, we define the associated skills: entities contributing to the process. We list the actions that can be activated with the possible exception thrown.

We use activity diagrams for our performances. They contain the events sent or received, the conditions of the transitions, the parallel workflows and exchanged business objects whose states are monitored.

When existing processes are described, it is possible to reconfigure to improve efficiency, improve flexibility. Further improvements are possible to provide a better level control and smoother operation and even reduce the execution time of processing. As an example of improvement, there is research into the causes of waiting by grouping tasks within a single activity or eliminating seizures or occasions of data.

2) Modelling approach

Our approach aims to extract the organizational aspects. It is important to analyze the process by respecting their borders. For this, we focus on objects involved in the process. Processes frequently collaborate within the same activities and it is important to specify the transactional aspects. This is specified as a string of treatments which obeys the rule of all or nothing. The scope should be as far as possible, as small as possible.

Of course, there arises the problem of transaction management long term which could several hours or several days. There is no question of pausing transactional locks on objects handled; the rollback is managed by a compensation mechanism.

3) Use view

An actor performs a series of transactions during his dialogue with the system. But restrictions apply: a scenario is not interruptible in business perspective view. In addition, a use case is single-player. Use cases describe the purpose of use. These are functional requirements that must leave the business information system in a consistent state.

Our method for identifying use cases is very simple. We first identify the actors and list the events and infer interaction decomposition systems. This view must be comprehensive to describe all interactions between the actors and the system. Each use case typically handles one main purpose. It is important to ensure that all use cases described

 ISBN: 978-1-61208-230-1

covers many transitions in the state machines of the major classes of the semantic model.

C. Physical distribution

To complete the functional requirements, it is important to locate users and geographic information systems. We add the technical requirements for equipment: technical guidance on network configuration requirements for the workstations. Other sensitive issues include competition within use cases and the constraints of scalability.

Nonfunctional constraints are taken into account as the degree of availability desired. Requirements related to quality of service, sometimes, bring the definition of categories of use cases. And a class level of service quality is associated with accessibility criteria.

Constraints related to the geographic dispersion are difficult because they require difficult technical choices. A site is described by its location, its capacity. He mentioned the players it hosts and the type of activities taking place there. Communication between sites is via the media. They should list them: communication network, transport, etc.

We use deployment diagrams, collaboration diagrams eventually. The collaboration diagram is often used early in the project to be within the information system and show the flow of information. Deployment diagrams are used in hand continuously from one project to another.

V. LOGICAL ARCHITECTURE MODEL

A. Approach

The logic model is used to specify and organize the service of SOA, based on semantic and pragmatic views. It defines the logical architecture of SOA will be derived in software components. Phase logical architecture of the system is similar to all phases of project management methodology. One of the rules is to minimize dependencies. Finally it is important to consolidate services related to business classes or use cases.

We chose to group services by field of business objects. We added a set of factories. These factories are the first level of urbanization. They correspond to the main structure of the information system. A factory has no interface but represents a logical division of classes. It involves important properties of the SOA. This multilevel structure organizes the data flow between the parties of the information system. Encapsulation manages the relationship between different components of the architecture: The level of services in terms of mask data.

Our methodological approach can be summarized. First, there is the structuring of the logic model in key areas. Then, there is the recovery of the semantic model and the derivation of detailed models, definition of complex data types. They are used to exchange information between services. They are grouped with the utilities. Then, we treat the analysis of use cases to discover the additional services. This involves grouping into packages. Finally, there is a detailed description of services (business and technical).

It is necessary to designate the services exposed to the outside world. For efficiency reasons, it is crucial to choose the type of invocation (web service, SOAP, XML, RMI, etc.). Transaction management also has an impact, especially for the resilience (the backup orchestration of context)

B. Structuring of the semantic model

Each main class of the semantic model is the heart of a logic component. The division into logical component follows the same structure as the division into business components. Dependence after a combination of the semantic model can be managed in several ways. First, it can be passed as a parameter of the service to remove a strong dependency. Secondly, it is possible to have data at the service orchestration. Do keep the dependencies in the types of data exchanged.

It is important to distinguish two types of services at the level of elementary components. On the one hand the services those run on a single component instance. These are the services associated with managing the life cycle. On the other hand the service for handling collections and navigation. These are services that work on sets to calculate a subset of data: search, sort, query, verification of existence.

The data exchanged are specified in detail because they are the pivot language that is used in the process orchestration. They reduce the number of service settings. The data related to a secondary class masked by a main class are managed by incorporation of a subtype in the main type. The data related to a semantic class of another component are managed by reference. They are retrieved by accessing a directory.

A logical component is described by an interface. All utilities are exposed to the outside. It also includes one or more internal data structure. Of course this interface is not accessible directly but through an access server. Optionally, a component can provide multiple interfaces for different missions.

C. Structuring of the pragmatic model

Each use case results in a Transactional service to validate a customer dialog. The transactional service logic starts a transaction that contains technical information transfer. This means that inspections are carried out with the use of a rules engine. If the checks are correct, the service validates the transaction and returns the information resulting from the transaction. In the pragmatic model, processes are described in terms of activity diagram. This diagram is attached either to the functional field or it's a package associated with crosscutting activities. The activity diagram is shown in the logic model and built to represent the functional area. It shows the services that keep coming into the process. Human interventions are indicated by actions that refer to use cases.

 ISBN: 978-1-61208-230-1

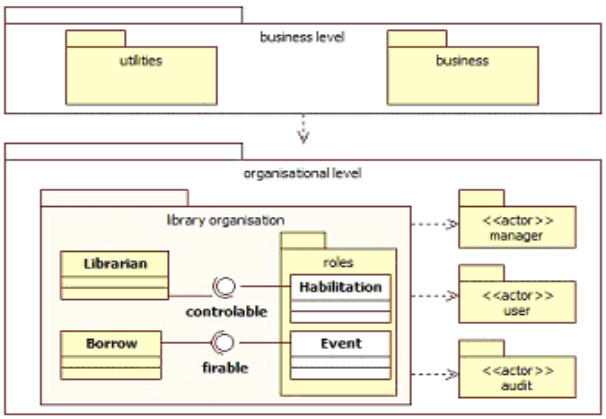

Figure 6. Structural diagram of the organization

In our previous example, organizations are run by factories logic. Organizations share the same core but can describe their operations as separate. In Figure 6, the decomposition of the system comprises both the context of use (organization, actor, ID), the informal context (set of information related to the trade).

VI. ARCHITECTURE AND ORCHESTRATION

A. Service specification

In service specification step, we may choose the signature and the names that will be present in the WSDL.. The parameters in reading, writing are prohibited. In an SOA, the transition is performed by parameter value to allow only based implementation web service. Additional parameters are the context, they provide information about the user context when the invocation. This information may influence the functionalbehavior of the service. They help minimize the number of services.

It is important to provide an exhaustive list of error codes. Non-blocking errors do not throw program level exception because these kinds of exceptions are catched and managed in framework level. We can provide services in signing a complex type that allows logging errors non-blocking.

Preconditions and post conditions can guarantee the conditions for running a service. They are based on the parameters passed as input. They are typically implemented by a direct appeal of the checking methods. This is a prerequisite for triggering of the main algorithm. We chose to externalize their code of the service implementation not the use of AOP (aspect oriented programming) [17].

Service quality is also specified. It is a guarantee of performance, availability and security. It relates to the average response time, the number of calls per second web service, the number of sessions, etc. We specify the end user monitoring: indicators and measures used. The documentation part of the services is important that a supplement should not be overlooked because life depends on it.

B. Technical aspects

Technical aspects that we addressed in our case study are the persistence, security, object-relational mapping, archiving issues, and the implementation of business rules and data architecture. Management communication was done by the web service call usually asynchronous.

We have implemented the business classes by POJO (Plain Old Java Object). We create factory for façade to delegate calls to the implementation classes. We distinguish the services handling a single component instance handling collection of services. These services must appeal directly to a data service that handles requests multiple instances of the database.

Queries performing joins on several business classes should be modified to remove dependencies between elementary concepts.

The process service calls keep coming. It is necessary to implement a facade since the methods have directly initiated the process. If several processes contain an identical set of activities, these can be managed using a process as implemented by a class in each package of integrated process that contains it. Implementation of this principle respects the principle of SOA but uses conventional technologies.

Processes are the orchestrators of calls to operations of business services. In terms of architecture, the process includes a presentation layer. A process is conventionally implemented as a component state full or using an orchestrator. In the first case, we must manage the execution context and make the system fault tolerant. We constructed an intermediate layer adaptation allows both to transform data and orchestrate existing transactions. The problem with JavaEE arises with the use of external transaction via tools such as SAP via JCA connector. We use a SOAP wrapper to trigger the transaction from operations.

We used the framework WISIF (Web Service Invocation Framework) from Apache to call the JCA connector. For security aspects, we wanted to make confidentiality and identification of access rights. The use of SSL is possible to exchange point to point but quickly becomes difficult with the spread and use of web services. The security management which is integrated directly within the SOAP messages [18]. The standard WS-Security OASIS framework provides a stabilized security manager. It enables strong authentication based on Kerberos ticket and is based on a W3C specification.

For transaction management, three aspects are taken into account with different frameworks. WS-Coordination provides a protocol to coordinate the actions of a distributed application (creation and propagation of context between the services). WS-Atomic Transaction defines transactions with a simple method of two-phase commit. Finally, WS-BusinessActivity can coordinate distributed activities with long transactions.

We used a middleware for the exchange of asynchronous messages. It has several properties: the ordonnancement messages, persistent messages in the event of service interruption, the integration of new components. All of our orchestration is made with the BPEL language. As

 ISBN: 978-1-61208-230-1

defined above WSDL, it can describe a collection of Web services. Activities can be combined with additional elements to form structured activities. For asynchronous calls, callbacks are defined by use of framework WS-Addressing.

CONCLUSION AND FUTURE WORK

We have presented our approach of orchestration definition. We have structured the design time in a sequence of model definition: semantics, logic, pragmatic. We have shown that relationships exist between them. These allow us to check and update our design. Then we have explained that BPEL scripts are derived and deployed into the business part of our multi-layer application.

Our future work will focus on extending our approach to other orchestration languages like CAMEL DSL [19]. Our goal is to enrich Java DSL's routing for managing dynamic mobile Participant that implements WS-Coordination. The participants are defined in functional layer; our approach offers a solution to adapt a functional definition to a generated Participant implementation to the constraints of the technical layer.

REFERENCES

[1] C. Szyperski, "Component Software: Beyond Object-Oriented Programming", 2nd Edition. Addison Wesley. 2002, pp. 139–150. ISBN: 978-0201745726

[2] J. Rumbaugh, I. Jacobson and G. Booch, "Unified Modeling Language Reference Manual," 2nd Edition, Pearson Higher Education, 2004, pp. 139-150, ISBN:0321245628

[3] Object Management Group. OMG Unified Modeling Language Specification, Version 1.4, 2001. http://www.omg.org/technology/documents/formal/uml.htm, retrieved: October, 2012.

[4] R. S. Pressman. "Software Engineering, A practitioner's approach". 7th edition edition, Mc Graw-Hill Education, 2000, pp 603-630. ISBN: 978-0071267823.

[5] P. Herzum, "Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise", Kindle Edition, January 2000, pp. 477-527.

[6] W. H. Inmon, J. A. Zachman, and J. G. Geiger, "Data Stores, Data Warehousing, and the Zachman Framework: Managing Enterprise Knowledge." McGraw-Hill, 1997, pp. 105-140, ISBN 0070314292.

[7] J. Zachman. The zachman framework for enterprise architecture. http: //www.zifa.com/, 1997, retrieved: October, 2012.

[8] TOGAF Version 9. The Open Group, 2009. http://www.togaf.info/togaf9/index.html, retrieved: October, 2012

[9] T. Erl, SOA Principles of Service Design, 1 edition, Prentice Hall, July 18, 2007, pp. 211-252, ISBN: 978-0132344821

[10] S. Hussain, B. Ahmad, S. Ahmad, and S. M. Saqib, "Mapping of SOA and RUP: DOA as Case Study," Journal of Computing, January 2010, pp. 2-4.

[11] S. W. Ambler, J. Nalbone, and M. Vizdos, "Enterprise Unified Process: Extending the Rational Unified Process", Prentice Hall. 2002, www.enterpriseunifiedprocess.com.

[12] C. Hofmeister, R. L. Nord, and D. Soni, Describing software architecture with UML. In Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1), San Antonio, TX, February 1999. , pp. 7-12.

[13] S. White, Using BPMN to model a BPEL process, BPTrends 3 (3) (2005) 1–18.

[14] A. Kleppe, J. Warmer, and W. Bast, "MDA Explained, The Model-Driven Architecture" Practice and Promise. Addison Wesley, 2003

[15] M. Elammari and Z. Issa, "Using Model Driven Architecture to Develop Multi-Agent Systems" the International Arab Journal of Information Technology (IAJIT), Volume 10, No. 4, July 2013, pp. 19-24.

[16] M. Peltier, J. Bézivin, and G. Guillaume, "A general framework based on XSLT for model transformations," In WTUML'01, Proceedings of the Workshop on Transformations in UML, Genova, Italy, April 2001, pp. 5-7.

[17] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J-M. Loingtier, and J. Irwin, "Aspect-oriented programming," In Proceedings of the 11th European Conference on Object-Oriented Programming, June 1997, pp. 3-5.

[18] S. Santesson, R. Housley, and T. Polk, "Internet X.509 Public Key Infrastructure Qualified Certificates Profile," http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I pp. x-y, retrieved: October, 2012.

[19] C. Ibsen and J. Anstey. "Camel in Action". Manning, 2010, pp. 113–122.

 ISBN: 978-1-61208-230-1