Adaptivity of Business Process

Charif Mahmoudi
Laboratory of Algorithms, Complexity and Logics,
Paris 12th University
Créteil, France
charif.mahmoudi@u-pec.fr

Fabrice Mourlin
Laboratory of Algorithms, Complexity and Logics,
Paris 12th University
Creteil, France
fabrice.mourlin@u-pec.fr

***Abstract*— Enterprise service bus is a software architecture middleware used for implementing the interaction between software applications in a Service Oriented Architecture. We have developed a strategy to dynamically manage business processes. Administrators of service bus need to reconfigure sites where the business processes are placed. This evolution has to be done during execution of service through the bus. We ensure the availability of process definition. Moreover, business process can also be autonomous. This means a process which is able to move from one site to another one, where the business process engine is installed. This provides another approach to design business process. With our "mobile process migration" template, we separate two concerns, on one side architectural features and on the other side business features. The business process can become mobile between two service busses and we improve the availability of business processes.**

Keywords-business process; BPEL; orchestration; middleware; message exchange pattern; code migration.

I. INTRODUCTION

Today, companies have tools to model and automate business processes. This type of tools allows formalizing the company's business rules to automate decision-making, that is to say, the branch of the workflow to choose from, depending on the context. The objective of this initiative is to achieve a better overall view of all enterprise business processes and their interactions in order to be able to optimize and, wherever possible, to automate up with business applications.

The lifecycle of a business process can be roughly broken down as follows: design, modeling, implementation, execution, control, and optimization. An approach of Business Process Management (BPM) is based on tools such as a tool for process modeling, tools support the implementation, a runtime loaded to instantiate processes, management tools and reporting. These reports show accurate and relevant indicators on the current deployment of business process definitions. Our first remark is on the lack of scalability of this deployment. Thus, the load of messages that flow through the middleware clearly shows an unbalance that affects the entire information system. So the first point is: how to adapt the workflow running.

A second remark is about the number of messages exchanged increases as a function of the initial placement of business process definition. Thus, a business process using local services is less costly in a number of messages than a business process using remote services. Blockings are also less numerous, and, therefore, the execution of a business process is more efficient.

This remark highlights the dependencies between two concepts, the location of business processes and its own definition. The designer should not consider his work in the placement constraints. In addition, the administrator cannot take into account all the dependencies of a process definition to find a better placement. Also, our second point is: how separate the two. These conclusions led us to consider an initial configuration of business processes is not satisfactory. This placement must be scalable over time to adapt to client needs. The implementation of this idea is described in this paper through a technical framework described subsequently. It allows the validation of the concepts presented here and provides a sample application.

The content of the paper is structured as follows. First, the following section discusses work related to our topic. In Section 3, we provide the definitions on which our work is based. Next, we describe the technical framework of our work. Finally, we provide a simple case study to validate our approach. We end with a point on the goals achieved and those that remain to be addressed.

II. RELATED WORK

The construction of information systems is usually performed by the department, each business building a subsystem adapted to its own needs and supported by heterogeneous technologies, rarely interoperable. To quickly meet the growing computerization of procedures, systems integration issues has emerged, and with them two questions:

- How to trigger in response to an event in a given subsystem, a treatment in another subsystem that is foreign?
- How to ensure consistency and spread data across multiple subsystems?

A number of technical solutions have been found to answer these questions. The implementation of these integration solutions is most often done on an opportunistic basis, to meet the immediate goals of a particular application. As these ad hoc solutions have been implemented, the problems of localization or global management have emerged:

 ISBN: 978-1-61208-246-2

- Flows have increased, sometimes redundant, and the chains binding techniques;
- Increasing the coupling systems brought its share of problems, synthesized by the concept of spaghetti effect;
- Organizations have to solve new organizational challenges. If chains of responsibility were clear for each business subsystem, what about the relations between these business systems?

Two broad categories of solutions have emerged: the ETL (Extract Transform Load) tools, to synchronize data from multiple systems, and middleware solutions, to ensure communication "real time" between heterogeneous systems.

A. ETL middleware

ETL tools provide synchronization, consolidation and spread of data between disparate subsystems. Schematically, they extract data from the master system to update subsystem, after a suitable transformation. Although they can operate continuously, ETL tools are rather intended to treat plarge data set in deferred time, they appeared initially to ensure the loading of data warehouses [1].

Their relative simplicity of implementation is their greatest strength. They also allow a first level of structure of system information, pointing to the owners for master data. Coupled to pivot formats, ETL tools allow avoiding the pitfalls of point-to-point and functional coupling between systems too narrow.

Unfortunately, the ETL approach is focused exclusively on the data, and provides only elementary business semantics. It therefore fails to solve the integration process, and more to meet the challenges of service-oriented architectures. Service orchestration is useless with that kind of tool. R. Kimball explains [1] that the notion of business process does not appear with this family of tools.

B. Network centric middleware

Middleware solutions provide a technical infrastructure mediating between two or more systems. Their historical role is to transport a message from one subsystem to another, with a level of coupling more or less important. Appeared in the early 80s, MOM (Message Oriented Middleware) has an asynchronous semantics: the client constructs a message and sends it to the middleware, which handles the routing to one or more target systems. Communication is split into two, avoiding the coupling technique of participants. The guarantee of message delivery is entrusted to the MOM.

For many years, MOM remained largely proprietary solutions, forcing each part to find out how to interface with the broker, and limiting the ability of integration environments and languages supported by the publisher of the solution. JMS, the standard messaging of Java, has partially lifted this constraint, and CosNotification, the CORBA notification Service has remained confidential [2].

The MOM also offers routing capabilities often limited, requiring efforts to important configuration, each route must be explicitly defined, making their implementation difficult on a large scale. E. Curry investigates the use of POSA (Pattern-Oriented Software Architecture) interceptor pattern [2]. This facilitates dynamic changes to the behavior of the deployed platform but its scope is limited to a local domain.

Despite their respective qualities, MOMs and ORBs (Object Request Broker) [3] remain highly technical solutions. They allow both the spread and integration of data processing, but the semantics of trade remains basically point-to-point. The client must know the format of the message he sent to third party systems, this functional coupling systems is rapidly becoming a nightmare for maintenance and operation, especially if extended to all of the information system.

C. Enterprise Application Integration (EAI)

A new class of middleware has emerged: the EAI, a hub and spoke architecture, as opposed to network-centric architecture of MOMs and ORBs, in which a central component mediates between the client and physical target. This central component takes over all low-level technical issues (location, availability, cache, communication, and transformation, interoperability through specialized connectors, audit, track, security or transactions) [4]. Like ETL, they are further able to provide data transformation in order to limit the functional coupling between systems, and apply sophisticated routing policies.

In this role of super-connector and mediator, the EAIs have more than a conductor: the EAI can host high-level business processes, aggregating treatments performed in several subsystems. R. Abate explains that Service–Based Architectures (SBA) transform traditional EAI efforts to the new level.

Despite their obvious qualities, EAI solutions suffer from their own nature:

- The protocol used for exchange and transport of messages in an EAI, is specific.
- The technology inside the EAI is specific also. Thus, application access is done through connectors still largely peculiar to each vendor despite attempts at standardization as JCA in the Java world (these connectors still often are very expensive).
- The data formats and data used in EAI is specific.

The EAIs became a too complex brick covering too many responsibilities in information systems.

D. Enterprise Service Bus (ESB)

ESBs come directly from EAI. Just check the list of major publishers of ESB to be convinced: Bea [5], Tibco [5], Oracle [6], IBM [7], Apache [8] are precisely those involved in the EAI. Embodying the architectural features of EAI solutions, ESBs focus on the functions of interconnect and mediation, and base this on a set of standards including [9]:

- The Web Services to manage synchronous communications.
- XML to define message formats
- JMS to send an asynchronous communication with MOM.
- JCA to connect to software packages and exotic systems (ERP, CRM, Main-frames, etc.).

Today, the ESBs are technology integration and intermediate application for implementing a service-oriented

 ISBN: 978-1-61208-246-2

architecture. But, they remain an elegant and sophisticated technical solution attached to the questions of inter-application integration. Their use does not guarantee success or even the reality of the implementation of an SOA. Their administration is also quite complex. When an administrator moves the definition of a service, the consequences can be unexpected. Thus, a high-level change cannot be directly applied because the runtime context can be considered with attention. M.T. Schmidt highlights that service orchestration is a key concept for the management of business unit into a whole distributed system.

E. Mobile Agent Platform

Mobile agent platforms have been proposed as a useful support for building distributed applications. They present interesting advantages, such as autonomy, flexibility, and effective usage of network bandwidth. Due to these features, they have also been considered as an enabling technology for mobile, wireless and adaptable computing. Nowadays, mobile agents are still an important focus of interest.

Mobility can be used as a way to move the code instead of moving data. This is essential when the data is very large or when safety prohibits any transportation. In our work environment, mobility is seen as a means of administration between sites. Code management in distributed systems needs this ability to be responsive to change. Using mobile agents, tasks requiring a lot of processing must be custom built to distribute the load between computers [10].

Ilarri, et al. [10] explain how code mobility can be used to manage computing resources across a network of computers. When a resource is not available, an adaptation is to move to another site where the computation can continue. Because the problem occurs at a precise location, its management is locally taken into account and a migration of context is done. We consider that this concept is translatable into another domain like software bus.

III. WORKING CONTEXT

The context of our work focuses on the management of SOA. These architectures are currently the subject of interest for many software engineering teams. These architectures are particularly interesting because they use open standards. They offer more possibility of applying new software standards and rules and to have new assembly. As part of our work, we use business services that we assemble to build orchestrations of services. As part of a distributed system, these orchestrations use services on remote sites. This brings new problems of availability of services [11].

A. Orchestration of business services

Web services are defined in two contracts: the data contract (XML schema for the operation signature [12]), and the service contract (WSDL description [13]). Our approach to building services is a classic one; it is based on the construction of a contract as the first step of the software Lifecycle (Contract First). This is a pragmatic and business driven approach, because it stresses what is expected of the service and not how it will be implemented [14]. We follow three steps for building a contract first Web service:

- The definition of service contract
- The writing of service endpoint
- The configuration of the endpoint

We adopt a similar approach to construct the definition of service composition. In other words, the coarse grain services are based on those of finer grains. We chose WS-BPEL [15] as orchestration language because it is well famous for defining business processes describing Web services interactions. Thus, we build a service oriented solution utilizing Web services with WS-BPEL, and we apply two phases:

- Build Web services as previously and then publish them to be utilized within a business process
- Compose the Web services into orchestration flows with WS-BPEL.

A first important difference between these levels is on the service status. If a service is said to be stateless, it is not the same for a composition of services regardless of the state. Thus the interpretation of a service composition requires consideration of a specific execution context. We use an orchestration engine to manage WS-BPEL scripts.

When we define an assembly of services, we depict the coordination by the logical algorithm into a WS-BPEL business process. Into a context of academic library, an orchestration is defined for the registration of a new member. The business algorithm schedules a sequence of steps: record the civility, record the profile, print member card, notifies by email the validation of registration. Each step is considered as its WSDL description.

So, we can create an orchestration to be used as a service within another, larger orchestration.

B. Use of business process with a bus

The orchestration of composite services in existing techniques is usually centralized. This is due to the features of participating services which are distributed and autonomous. A centralized orchestration model has several drawbacks with respect to scalability and availability [16]. Because of Web service characteristics are highly dynamic, autonomous and distributed, we believe that orchestration of services can be interpreted in a more dynamic way.

To make a more dynamic interpretation, it is essential to have several BPEL engines. These engines are distributed over network, for instance at least one per service bus. Thus, we define a start and arrival point. In addition, we need a way to communicate between these two points. In our context, the notion of message is known. Each operation on a BPEL description can be seen as a particular message transmitted between one or more engines. As an example, interpretation is a consequence of an input message, migration also. We consider that an intelligent routing of these messages is able to provide the dynamic interpretation. This is done by a software bus.

We decided to use an ESB because it is the technical frame where all parts of our work can be gathered: several BPEL engines, binding components and service engines (Figure 1). But monitoring tools can be added and other ESBs which can be connected into a larger distributed system. First, we started by designing a move operation of

 ISBN: 978-1-61208-246-2

business processes from one site (called Group1) to another one.

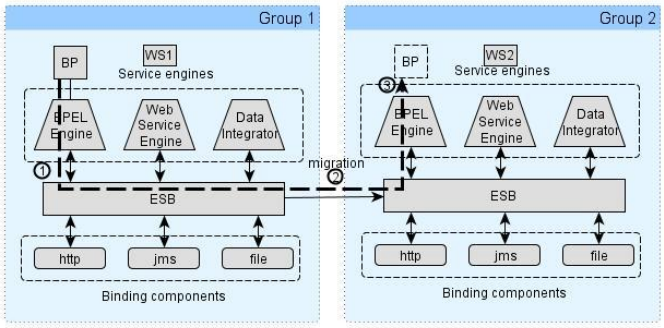

Figure 1. Relation between two ESBs

This technical operation can be considered as a kind of management operation into a large distributed system. Our nominal scenario is triggered by an event which highlights (1) that a business process called BP needs to be moved from Group 1 ESB to Group 2 ESB. Because both ends of the exchange are known, a route can be built (dashed line) and BPEL script can be downloaded (2) from a local repository to Group2 ESB. Then, this script is activated through the local BPEL engine (3). Finally an interpretation can be done by another engine.

This approach has drawbacks. In terms of safety, it is necessary that the issuer of travel demand (on a BSE Group) has a role as it has the permission to publish a definition of business processes on BSE Group 2. These non-functional constraints are taken into account in a real context.

To manage dependencies, it is important to check that the definition of business processes that is moved contains no dependence on the BPEL engine that used it originally. If this were the case would cause a shift in an execution failure in the Group 2 sites.

C. Administration of services into an ESB

The description of this management operation highlights the steps of routing. We thought a good way to automate this process is the definition of a dedicated BPEL script. Thus, we define a set of operations on the business process definitions. They focused on adding, moving, copying, deleting BPEL definition (Figure 2). The design of these operations requires an understanding of the functioning of an ESB to make the most of the modules already active. If the administrator is considered the trigger of management actions, it is the purpose of simplification, because other actors may be able to: an event configuration of ESB, a demand from the business process itself, etc.

Compared to the various modules that make up an ESB, the steps for interpreting a functionality such as "moving a business process', requires the use of the routing module, the XML transformation module, the message queue manager and the BPEL interpreter. In the scope of this document, we only interact with BPEL script, which are not under execution. If it is, such operations like “move” are postponed until the end of the execution.

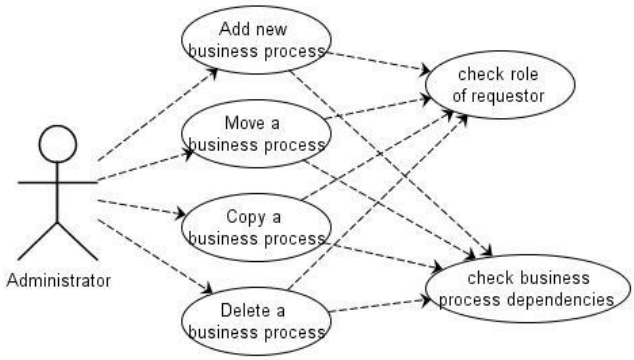

Figure 2. Use case diagram of BP management

Figure 3 describes the activity diagram of the move operation. After selecting a business process definition into the repository of a BPEL engine, its definition is parsed to detect conflicts. This operation is realized by applying technical rules onto the BPEL script. Then, the route builder is invoked to define a new route between source engine and target engine. This route is engaged and can be used for the migration of business process. Because controls have been done successfully, the BPEL can be downloaded from the intern repository into the working directory of the new BPEL engine. By the end of this action, the process has to be activated.

Then, the route is used to transfer input messages from source site to the target site. This step can be optional if we consider the operation as a clone operation to divide the traffic by two. Finally, the business process definition can be instantiated by the target engine. This instance can accept input messages previously sent to another site. The state of this instance is managed by a new BPEL engine. It may be suspended, in which case it is counted as suspended and not active. If an error occurs that does not cause the process to complete, but requires attention, the process is counted in error instead of active. If a process is terminated with the exit activity, it counts as terminated. As shown in Figure 3, only the migration of the definition is described and not the treatment of the input messages which is the normal process of a BPEL engine.

The activity diagrams of other management operations follow the same schedule even if the core actions are totally distinct. Preconditions of all operations are quite similar and there is no invariant about their applications. This set of operations is the basis of our approach of adaptability of business process. Our analysis is completed by requirements about no functional properties which are essential in a large distributed system. They specify global constraints on how the software operates: no blocking, authentication with role names, asynchronous message exchange, etc. These constraints are guidelines for our realization. No functional properties have a global nature, in opposite to local effects of functional requirements. In that case, the modification of a single bundle of the distributed system may affect the integrity of the entire application with respect to a particular

 ISBN: 978-1-61208-246-2

no functional feature, such as asynchronous communication. In next section, we focus our choices with technical aspects.

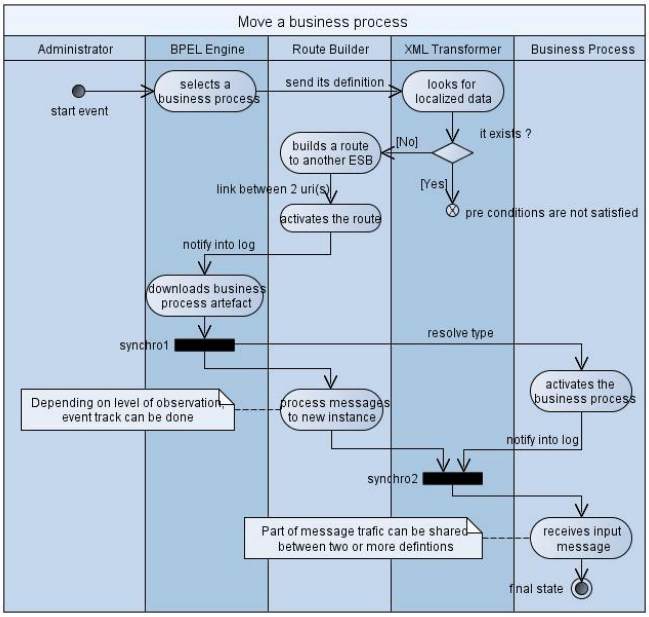

Figure 3. Activity diagram of "move business process"

IV. TECHNICAL APPROACH

Our past experience in distributed systems is based on pragmatic approach. Also, we build a prototype, because it is a true validation of our ideas. In the context of this document we use an ESB called Apache ServiceMix because it is a reference in the world of open source solution [17] and also because we participate to the evolution of this Apache project.

A. Description of technical context

Our chosen ESB allows building ESBs clustering and also linking several ESB through message queues. Several software architectures are possible and we used a cluster for the case study. The container can provide failover strategy and a configuration can be set at load time. The deployment of service units and binding components is dynamic and the Lifecycle of business objects can be managed through a programmatic API.

ServiceMix is often coupled with Apache ActiveMQ for message queues management, Apache Camel for the route management, Apache CXF, as a web service engine and Apache Ode as BPEL engine. Because all these modules are written in Java, a JMX console is used to display attributes and operations of managed Beans. These are the main modules of this bus but they rely on OSGi server called Apache Karaf which is the kernel of the ESB (Figure 4).

Thus, our project is composed of five elements which will be exploited by ServiceMix bundles.

1. The context description is an XML file which used to create all objects of the scenario. It also injects code to setup and to configure them.

2. Because, all communications are asynchronous, each part of our solution is equipped with an input message and an output message queue. These message containers are defined by a URI (Unique Resource Identifier). A URI can be an end of a route.

3. A set of routes which contains at least one route from a source business process to a target which is the business process after its migration. Both are identified by a URI.

4. A BPEL script which defines the migration procedure. It can be duplicated onto all BPEL engines of the distributed system. Another solution is to define a migration procedure per BPEL engine. This can be useful whether business processes have preconditions before moving.

5. A set of externalized rules which checks whether a business process can be moved from one engine to another one. Because a BPEL definition is first of all an XML stream, a rule is written with XSL-T language. This allows changing dynamically the rules even if the bus is running. The goal of these rules is to express if a BPEL definition depends on local resources, such as low level API or specific codes.

All the modules are assembled into an artifact which is deployed into the input folder of the ESB (ServiceMix). A first observation of the log console allows the administrator to understand if the XML descriptors of the artifact are valid. Next, we explain the deployment step and the role of message exchange.

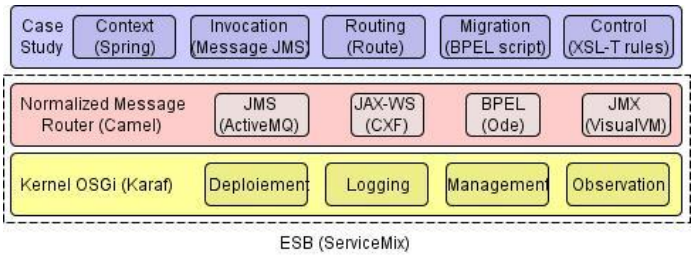

Figure 4. Layer description of our project

B. Role of normalized message

The normalized message router (NMR) is responsible for mediating messages between all the modules which are deployed into the ESB. The deployed modules do not exchange messages directly between each other. Instead, they pass messages to the normalized message router. The role of the router is to deliver the messages to the right endpoints. All functionalities are declared through its endpoint. Endpoints provides clients with access to business process. It is possible to define one or more endpoints for a service by using a combination of relative and absolute endpoint addresses.

Also, when a client sends a request to a business process, it is first received by a binding component. In our context, a client could be a traditional application, a tool like SOAP-UI or another module deployed into the ESB. The binding components are used to provide transport level bindings for the deployed processes. The normalized message model

decouples the service client from the service providers. The message format is defined using WSDL, which describe the called operations.

A normalized message is a generalized format used to represent all of message data passed through the NMR. It consists of three parts: meta-data and properties, payload, attachments. The binding components are responsible for normalizing all of the messages placed into the NMR. Binding components normalize messages received from external clients before passing them to the NMR. Messages sent across the NMR are not persisted anywhere but we can modify the process to write these to a database using the Data Base binding component or otherwise.

Then the message is delivered to a service unit like a BPEL engine or another module deployed into ServiceMix. Service units can be grouped into an aggregate deployment file called a service assembly. This file includes a deployment descriptor that indicates the target component for each service unit.

C. Functional requirements

The BPEL engine treats requests and instantiates process as needed except if there is rules which need management operations. In our scenario, the called process is used to register conference inscription. We added a rule that limits the number of instances. The value of this limit is one, because this triggers easily a move operation of business processes. So, this triggers our BPEL process called MAH process, this is the main line of the management operation. As mentioned in Figure 3, first a web service applies a sequence of rules to build a diagnostic. In this example, the answer is affirmative.

The first part of the MAH business script as shown in (figure 5) the dependencies of the definition with two partner links which are mentioned previously. Because all these partner links are valid, the evaluation can be done by the BPEL engine.

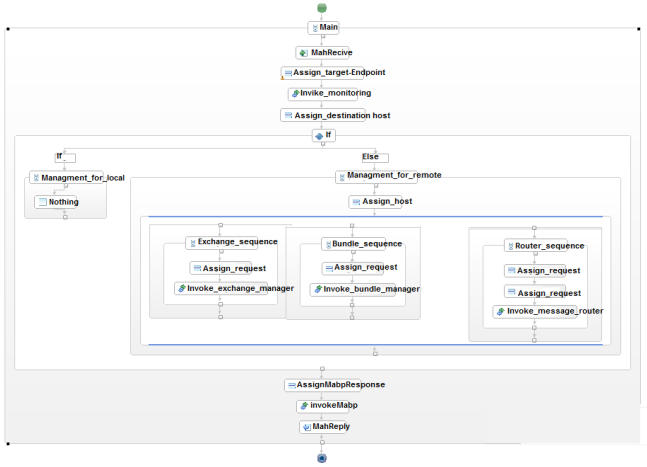

Figure 5. Design of the migration business process

V. CASE STUDY AND RESULTS

Our case study focuses on the migration of a process making two calls to web services by "localhost" because it is assumed that for security constraints, the server only accepts local connections. The process will make an initial local call to retrieve the result of the invocation of the first web service. The process will then migrate to the host which exposes the second web services in order to invoke this web service locally. The core of the definition of BPEL processes MAH is the red line of the script: each sub-process manages a step towards translating the script from one node to another. Thus the interpretation of the orchestration is not monopolized by the BPEL engine since the MAH manages the mobility aspect.

A. Evaluation of our test case

The evaluation of the sequence of actions starts with a message about a business message to move.

```
<receive name="mahReceive"
    createInstance="yes"
    operation="MAH"
    partnerLink="MAHPartnerLink"
    portType="mah:MAHPortType"
    variable="mahRequest"/>
```

Figure 6. Test BPEL request

This request identifies the process to move and also to check the technical rules such as the initiation of remote communication with the MAH as a prerequisite. Multiple Web services are used to prepare the management operation; they are exposed by the MAH: they can transport the process to enable message routing normalized and manage ongoing exchanges with the process.

Since the first invocation does not require migration, the role of the MAH will be transparent. The second invocation will require the execution of various technical tasks mentioned in the previous paragraph since the target web services is identified as external to the host.

B. Observation of the migration process

Tracking the evolution of the migration is essential to decide whether controls are applied at the right time. Moreover, it allows also summing up when administrator wishes it to know the path of a business process during a period of time:

```
  <bpel:invoke name="Invoke_monitoring"
operation="GETINFO"
    inputVariable="MonitoringRequest"
outputVariable="MonitoringResponse"
    partnerLink="MonitoringPartnerLink"
portType="mo:MonitoringPortType" />
```

Figure 7. Service invocation

 ISBN: 978-1-61208-246-2

Each step of the migration process is monitored by observation services which write into log files what happens during the migration process.

C. Impact on the other management operations

When the migration business process is evaluated by BPEL engine, a set of tools is deployed as mentioned previously. The ESB's console (Figure 8) show all the elements which are built to achieve the move of a business process.

The following figure displays a route module, several web services (these elements are outlined in red color), binding component also.

Of course, this figure shows a snapshot about what it is currently deployed by the end of the migration process. Some elements could be reused for another migration but it is not the case in this case study.

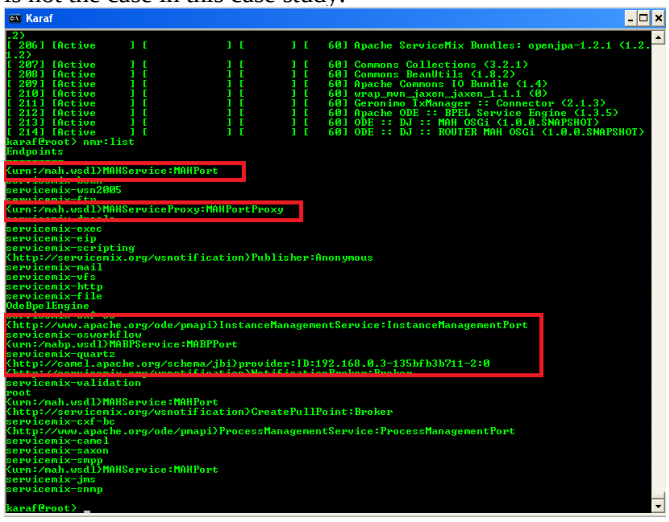

Figure 8. Console of ServiceMix ESB

VI. CONCLUSION AND FUTURE WORK

To sum up, we have described the life cycle of our approach of business process management. First, we have shown how it is possible to move a description from one server to another one into a cluster of servers. Then we have detailed how we have prototyped it as a specific BPEL process which is a conductor of a reconfiguration of the client business.

Next, we gave results about our experiments. Because this work uses dynamicity, it is not easy to highlight mobile feature, but we wanted to stress that all steps of our approach are taken into account. This validate that our approach is useful and also that ESB are servers which can exploit mobility as an ability to do an adaptation during execution.

We have realized other management operations but all these operations operate on business process definition and not on process instantiations. Also, we are working now on the mobility of instances of business processes and how to move an execution context without perturbation. We will focus on extending our approach to other orchestration languages like CAMEL DSL [18]. Our first goal is to enrich Java DSL's routing for managing dynamic mobile processes. Camel is based on the separation between aspects "definition" and "execution" of business processes. Camel business processes are based on a sequence of EAI, it allows us to add a new pattern "Migrate" which may be declared in Camel DSL and will be provided at runtime by a mechanism of self-transfer of business processes to the target host. Our second goal is to construct a formal specification in "pi-calculus" [19] of Camel's engine using the approach based on business specification [20] to highlight the mobility aspect added. Thus, the result demonstrated that the reduction leads to a transparent migration from an external point of view.

REFERENCES

[1] J. Caserta and R. Kimball, "Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data", 2004 , ISBN-10: 0764567578

[2] E. Curry, D. Chambers, and G. Lyons, "Extending Message-Oriented Middleware using Interception", presented at Third International Workshop on Distributed Event-Based Systems (DEBS '04), ICSE '04, Edinburgh, Scotland, UK, 2004.

[3] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers," *Computer Communications*, vol. 21, pp. 294–324, Apr. 1998.

[4] R. Abate, J. P. Burke and P. Aiken "Eai with Service-Based Architectures", 480 pages, John Wiley & Sons (Oct 2002), ISBN: 0471415154

[5] I. Charlesworth and T. Jones, "The EAI and Web Services Report", EAI Journal, 2003, pp. 11-18.

[6] J. LEE, K. SIAU and S. HONG (2003) Enterprise Integration with ERP and EAI. Communications of the ACM 46(2), pp 54–60.

[7] L. Gavin, G. Diederichs, P. Golec, H. Greyvenstein, K. Palmer, S. Rajagopalan, and A. Viswanathan, An EAI Solution using WebSphere Business Integration (V4.1), ISBN: 0738426547, 2003

[8] P. Kolb. Realization of EAI patterns in Apache Camel. Student research project, University of Stuttgart, 2008.

[9] M.T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. "The enterprise service bus: Making service-oriented architecture real". IBM Systems Journal, 44 (4): pp 781–797, 2005.

[10] S. Ilarri, R. Trillo, and E. Mena, "SPRINGS: A scalable platform for highly mobile agents in distributed computing environments," in Fourth International WoWMoM 2006 Workshop on Mobile Distributed Computing (MDC'06), Niagara Falls/Buffalo, New York, USA. IEEE Computer Society, June 2006.

[11] J. Waldo, A. Wollrath, and S. Kendall. "A Note on Distributed Computing". Springer Verlag. 1994.

[12] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana: Web Services Description Language (WSDL) 1.1,15 March 2001. http://www.w3.org/TR/wsdl, retrieved: December, 2012.

[13] J. Farrell, H. Lausen "Semantic Annotations for WSDL and XML Schema", W3C Rec., Aug. 2007.

[14] S. Loughran and E. Smith. "Rethinking the Java SOAP Stack". May 17, 2005. Copyright © 2005 IEEE Telephone Laboratories, Inc..

[15] Web Service Business Process Execution Language Version 2.0, Working Draf, July 2005, OASIS Technical Committee, available via http://www.oasis-open.org/committees/wsbpel, retrieved: December, 2012.

[16] Q. Chen and M. Hsu. "Inter-Enterprise Collaborative Business Process Management". In Proc. of 17th International Conference on Data Engineering (ICDE'01), IEEE Computer Society, April 2001, pp 253-260.

[17] B. A, Christudas, "Service Oriented Java Business Integration" (1st ed.), Packet publishers, (August 13, 2008), pp. 436, ISBN 1847194400.

[18] C. Ibsen and J. Anstey. “Camel in Action”. Manning, 2010, pp. 113–122.

[19] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I and II. Volume 100 of Journal of Information and Computation, 1992, pp 1-77.

[20] C. Mahmoudi and F. Mourlin. International Conference on Software Engineering Advances, thinkmind , Lisboa 2012, pp 197-204.

 ISBN: 978-1-61208-246-2