Web service orchestration driven by formal specification

Charif Mahmoudi[1], Fabrice Mourlin[1]

[1]LACL, Computer Science department, Paris-Est University, 61 avenue du Général de Gaulle, 94010 Créteil Cedex, France, EA 4913
cm@ramses.fr, fabrice.mourlin@univ-paris12.fr

Abstract. When set of basic web service is built, the next step is to create more complex one. A programmatic approach uses declarative language such as BPEL. This kind of representation is verbose and needs assistance for the creation of business process. We defined a generative strategy leading by formal specification. Because, web service composition languages use standard operators like sequence, choice and parallel; process algebras are right candidate for their design. We use higher order pi calculus for specification of orchestration and consider it as a basis for generating BPEL skeleton. After enrichment of BPEL definition, we interpret it by mobile agents. They play role of a conductor which evaluates a part or a whole BPEL definition. This mobile engine reduces message traffic by replacing message transfer into local message operating. We use mobile agent as distributed conductor of business processes.

Keywords: web service, orchestration, composition language, process algebra mobile agent,

1. Introduction

Web Services mean many things to many people. Today, a lot of developers have tried to expose web services on their local network. Their first example is often to build a simple web service that generates a response based on information received from the client. But also, it shows how to consume various existing framework web services, including existing services such that computing service (Choleski computation) or geographical service (a map of a next trip in Edinburgh).

Web service programming is easy and straightforward. They must typically be constructed as basic building-blocks of modern programming practice. For example, a layer of web services is built for allowing interaction with other software such as a persistent layer or a business layer. Then, this web service application programming interface is considered as a technical interface that facilitates communication between two tiers. One of its main properties is about the use of XML language as request language. Also, more complex code can be created on that layer, even more complex web services, or business processes.

Because, their development cycle is easy to manage, extensive strategy is adopted into concrete project to apply this kind of development. As a result of this, designers are looking for more and more features, and they are trying to do everything by using Web services. Problems occur with arrangement of web services. It is not so easy to find a global distributed organisation. More complex, is the use of run time context through all web service calls. This problem is often linked to the management of a computation state into a business process. By definition, Web services are said to be stateless. There was no notion of managing sessions in the Web service world. But a business process has to manage a coordination context and an evaluation cursor.

Now development frameworks have changed and programmers cannot develop an advanced application without managing state. As a best practice, a handler or transport should not keep any instance variable because that breaks the stateless nature of web service. Managing sessions directly causes an increase of the memory footprint and to slow down performance.

More complex is the transaction management. Their specifications define mechanisms for transactional interoperability between Web services domains and provide a mean to compose transactional qualities of service into Web services applications. It describes an extensible coordination framework with short duration, ACID transactions and longer running business transactions. Again development frameworks enable Web services to participate in a distributed transaction as prescribed by the WS-Coordination and WS-Transaction set of specifications [1], but technical limits are set about transaction management. Very few frameworks implement these specifications and specific middleware are often used. This is a strong limit of a distributed system and a barrier to scalability.

A good example would be monitoring applications where users log in into the system, local activities, and log, out. If we try to map this example into a Web service, we create a sequence of call. First, the user logs in into the system (invoking a login method). Then, the user invokes local work (invoking some operation on his local account). Finally, the user logs out (completing the transaction). It is easy to understand that the three operations described above are interrelated, and the same user does all three invocations. So, it means that someone needs to keep track of the user, and another has to keep track of the user data throughout methods invocation. This implies the need for session management. Because, all these calls belong to only one activity, transaction is also needed. In that context, we depict a software transaction of web service invocations.

Transactions are assumed to be defined in the context of a more global representation of this sequence of web service calls. This is commonly called business process. A business transaction may be unpredictable and can cost a lot of time. Thus they may be unacceptable to lock resources exclusively for such long while. This means that this new concept introduces new complex problems. This involves analysis of the description of business process and detection of some properties. Tools appeared for modelling them via new notation based on workflow pictograms. Others use more technical notation like BPMN (Business Process Model Notation) or XML language with BPEL (Business Process Elementary Language). All notations gather a large set of information. Its collection is done during the progression of the design time. They will be used not only for the orchestration of all the web service but also for different kinds of test such as pragmatic unit testing, deployment tests, functional testing or integration testing. Also, it appears that there is need to get a specification which groups all features useful for orchestration and test. This document describes our contribution to the domain of business process specification.

We divided this document into several sections. Next one is about the role of web service into business process definition. Next section is about the process of construction of a business process and it is followed by reasons to formally specified business process. Then we propose higher order process algebra and we precise its expressive power. Finally, we use this formal notation as a pilot to generate BPEL skeleton and we explain the role of designer who completes orchestration script. By the end, we show our BPEL evaluation strategy by the use of mobile agents?

2. Use of Web services

The main reasons for using Web services are frequently to gain interoperability among distributed applications that span diverse application servers. This interoperability is based on a cross language (XML), this data model helps to develop heterogeneous distributed applications [2]. In addition, the accessibility of applications is improved through firewalls using HTTP protocols.

Because Web services are accessed using standard Web protocols, such as XML and HTTP, the large and heterogeneous applications on the Web can automatically access Web services, solving the ever-present problem of how different systems communicate with each other. These systems might be written in a variety of programming languages, such as Java, C++, or Php. As long as the application that provides the functionality is packaged as a Web service each of these systems can communicate with any other. This allows developers to expose resources onto HTTP network like computational results or photographs of many weather phenomena.

Web services simplified model for creating and connecting distributed applications across the web in the form of services. From a more technical observation, Web services are XML representations of objects, messages, and documents designed to interact over the web to enable application integration. Also, when two different information systems have to communicate; web services can be considered as data wrappers and adapters. More interesting, if both information systems do not know each other; Web service applications can be published, found or invoked dynamically. Also when two partners into an academic project want to provide a common interoperability layer between disparate platforms; they build Web services to connect with each other based on their own business. The benefit will come in the form of enhanced user experience as a much bigger set of services are provided and managed to end users.

Students training are also an application domain of Web services. When a university, called Paris12, offers to remote students, Web services for training management; this is not only a new way for providing teaching material for oversees students. This approach allows others universities which share same projects, to adopt a part or whole solution of set of Web services. Use of XML data exchange allows adapting data formats between Web sites. When the web service set has to be integrated into a more global offer of another university; the composition of Web service has to be solved. Several solutions can be considered: building new Web services based on previous ones, selecting a language for composition expression, like XLANG, WSFL, XPDL or BPEL. Thus, richer Web services can be exposed through other servers and traffic could become more important on initial university web server. But if the number of students increases at given university (Paris12 university), this solution does not allow others universities to increase their number of their students. It means that another use of this web service set could be the importation of their definition contract and its redefinition. WSDL definition is used as a declarative interface which can be freely copied and used to create a local implementation of the Web services.

3. Composition of Web services

In industry terms workflow and document management systems are used to describe how components are connected to build complex business processes. They describe the flow of work between several software organizations. Such software may include partner applications (legacy systems) of local or remote software built in different technologies and even people to perform certain tasks.

With the introduction of richer Web services through "web services composition" or "web services flow" [3]; Web services have been used to describe the composition of web services in a process stream. This has sped up research on problems of coordination and interoperability of web services. The solution proposed by most of this work is to model a software process.

This process manages the coordination of acitivities and operations of this software. This research around the coordination of web services has created two new words: Choreography and Orchestration of Web Services. The choreography is modelling the sequence of message exchanges between web services and conditions under which these messages are exchanged between clients, suppliers and partners. Choreography is typically associated with the exchange of public messages between web services, whereas a business process is executed so centralized. In recent work on the choreography, we find the language WS-CDL (Web Services Choreography Description Language) is an XML-based language for describing peer collaborations between the participants and the exchange of messages between participants in the objective to accomplish a common goal in business logic.

The service orchestration defines the sequence of services according to a predefined model, and run them through scheduling scripts. These scripts are often represented by business processes or workflows inter / intra-enterprise. They describe the interactions between applications by identifying the messages and connecting logic and invocation sequences [4]. Orchestration describes the strategy in which web services can have exchange with other through message passing. It includes the business logic and execution order. These interactions can cover applications and also organizations. The result can be a model of long-term process, with transaction, and multi-step evaluation.

An important difference between orchestration and choreography is that the orchestration provides a centralized observation which means that the process is always controlled from the view of a business partner. However, the choreography offers a collaborative coordination. It describes the role played by each partner involved in the whole application. Our contribution aims to change the centralized aspect of the business process evaluation to a more distributed view of business logic. The dependencies of partner processes do not play same role depending where the evaluation is.

The main language for the coordination of web services are: WSCI (Web Services Choreography Interface), BPML (Business Process Modelling Language), WSCL (Web Services Conversation Language), BPEL4WS (Business Process Execution Language for Web services) BPEL4People which is the WS-BPEL extension for people, proposed in a joint white paper by IBM and SAP [7].

WSCI is a description language based on XML [9]. Its role is to define the message types exchanged between Web services belonging to business process. WSCI allows to declare the dynamic interface of the Web service belonging to a given message exchange. The dynamic interface definition contains the operation signatures. These declarations are added to WSDL and SOAP streams, to give description of the types of messages.

BPML is a meta-language that was developed by the organization BPMI.org (Business Process Management Initiative) [8]. BPML allows defining a business process as a sequence of simple, complex activities and processes including interaction between partners in order to achieve a business goal through the use of events and messages.

The language WSCL describes the exchanges from the business logic level. This is low coupling of public process definition of a web service. The definitions of WSCL conversation are XML documents and can therefore be interpreted by Web services infrastructures and development tools [10]. A WSCL stream can be used in conjunction with other service description languages like WSDL to provide formal contract protocol for abstract interfaces, or to specify the abstract interfaces implemented by a concrete service.

The language BPEL4WS (Business Process Execution Language for Web Services) or simply BPEL subset, is a specification from IBM, Microsoft, and BEA. It replaces the previous specifications of Microsoft XLANG and WSFL (Web Services Flow Language). The BPEL process model represents a layer on top of WSDL. It defines the coordination of interactions between the process instance and its participants. Processes in BPEL export and import remote or local functions using web services interfaces only. BPEL has the characteristics of a block-structured language of XLANG with typed variables and the specific graph evaluation of a direct WSFL.

BPEL can model two types of process: abstract and executable. The abstract process specifies message exchanges between the various parts, without specifying the internal behaviour of each. The executable process specifies the evaluation order of activities constituting business process, the partners involved in the process, messages exchanged between these participants. It details also processing errors and exceptions and defining the behaviour in case of errors or exceptions [11]. The great advantage of BPEL is the ability to describe the interaction between the business logic of individual organizations through web services. The elements of BPEL process are: the partner links, the variables for the execution context, and activities about data.

Several industrial implementations of BPEL currently exist: Zenflow software, ActiveBPEL (Orchestra), ODE (Orchestration Director Engine, Apache); an Oracle BPEL Process Manager which offer a comprehensive infrastructure for creating, deploying and managing BPEL business processes. These tools are BPEL interpreters; they read BPEL scripts and evaluate business process. The currently accepted version of the BPEL specification is version 2. It is expressive but not very intuitive. Moreover its XML representation is very verbose and its many, rather advanced constructs are not easy to use by end users and even simple things can be implemented in several ways. Tim Hallwyl, Fritz Henglein and Thomas Hildebrandt have worked about a standard driven implementation of BPEL 2.0 with extensions.

In a typical BPEL scenario, a business expert or analyst of a company would use a BPEL Designer, with a Graphical User Interface and define the business process. This motivates the need for a "higher level" language allowing formal verification and from which one can automatically or semi automatically generate BPEL code. A good translation from the formal language to the BPEL process (BPEL 2.0 for instance) must reflect as much as possible intentions of orchestration conductors and the work flow constructs.

4. Specification of orchestration scenario

The orchestration strategies are presented as a set of mechanisms for construction of a new service (called composite) in an application composed of all services achievable. Variables are defined in level orchestrations that allow for sharing of information between invocations.

The new notations, associated with orchestrations, are designed to help maintain a loose coupling, ease of reading codes and reuse of composite services or not [12]. Without the orchestration, service composition is defined by a new service which reference to previous covertly sees in some cases to redefine the services themselves to express the interactions with the services required. This does incremental programming would not unanticipated.

In recent years, the couple BPEL / WSDL has emerged as the standard used for describing and orchestrating services. WSDL allows specification of interfaces and methods and their parameters in terms of inputs, outputs and exceptions. The declarative language BPEL is the most used for business process definition. It is implemented and therefore validated par users. However it is a very low level language, written in XML and WSDL-based, it is very technical and quite difficult to pick up even using graphical editors.

This couple is necessary because there are currently many tools to make creating Web services and processing sequences available such as automatic generation of client and server source code in almost all programming languages or graphical editors to advanced creating effective BPEL descriptions.

All these technologies are now mature and solve the many technical interoperability problems present in the SOA platform. Indeed, the work of software architects of service-oriented platform is now greatly simplified by these standards. However all these technologies are not trivial and requires considerable technical skill in order to benefit. Administrative tasks of a typical SOA platform therefore remain restricted to technical experts.

The most promising path to simplify, automate see some activities in order to make them accessible to a non-technical expert is the use of semantic descriptors. To meet the needs of simplifications in the creation of processing chains in SOA platforms, we chose a methodology based on formal specification. Process algebra such as Pi calculus [13] shared same concepts as BPEL language: sequence, choice, parallelism, etc. Recently, many researchers have exploited the studies on process calculi as a starting point to define a clean semantic model and lay rigorous methodological foundations for service-based applications and their composition. Their objective is to prove properties about service composition and to detect unacceptable behaviour. Property proof is of course an important use of formal language. But it can also be a pilot for generating a partial BPEL description and upper WSDL definition. Such formal specification is a way to declare service composition without technical details but with business logic. It is also an easy way for separating processes and exchanged message. Message transformations are closed to term unification or rewriting rules. Then it is XSL-T transformations can be computed from formal description about exchanged data.

All these considerations help to choose a formal language which is expressive enough to define data types, process type and also operations on these types. Thus, these constraints bring a selection of process algebra with set of operational semantics.

5. Orchestration specification with higher order π-calculus

Process algebra is often seen as a worthy descendant of no deterministic automata theory. The main difference is that nowadays one is interested not merely in the execution traces of specific automaton, but also in the behaviour of systems of communicating automata. Behind process algebra, there is an algebraic theory to formalize the notion of concurrent computation, best exemplified in CSP and CCS. These first algebras are evolved and new ones appeared with taking into account new requirements such as mobility and fault tolerance.

5.1. Higher order π-calculus

The Pi-Calculus was invented by Robin Milner. It can be considered as a programming language theoretical. This language plays in the field of parallel and distributed computing a similar role to that of λ-calculus in classical computing. The objective of the π-calculus is to study the important concepts in parallel programming and distributed to isolate the key mechanisms and reformulate the redundant mechanisms in the form of macros constructed from basic mechanisms. It allows also defining high-level concepts such as behaviour of a program (useful to provide assurances about the absence of errors), equivalence of two programs (particularly useful for writing compilers or garbage collection), etc...

The π-calculus is also able to control the construction of skeleton of declaration or programs. There are several variants of the canonical π-calculus. We use in this document, the polyadic asynchronous higher order π-calculus with replication and guarded actions [14]. In the π-calculus the simplest form of entity is a name. A name is regarded as the referencing convention for a channel, through which processes can communicate.

The action $\bar{c}\langle a\rangle$ in this system has the potential to output (reply) the channel named a. This channel must be known since the action is trying to communicate it, hence it is using the channel name a. A variable represents a placeholder for a channel which is currently unknown, but has the potential to become known.

Symmetrically, the action $c(x)$ has the potential to input (receive) a channel, and since it is not yet known which channel it will receive, then the variable x is used. Both actions play a role of start and end elements into a business process. At the beginning, a process receives information which will involve a set of statement. By the end a response will be sent to the caller if necessary.

An output action will communicate with another process, by sending the tuple of channel names y_1, ...y_n, through the channel c. When this is completed, the execution of the process continues as necessary.

An input action communicates with another process through channel c, where it receives a tuple of channel names. As we have already described, we have a tuple of variables, y_1, ...y_n, awaiting to receive the channel names from another process during communication. Thus these variables are substituted to channel names throughout the process, once these names are received. Because, data are typed, it is essential to define as precise as possible message types. If π-calculus actions have a direct correspondence into BPEL language, data types have impact into WSDL definitions.

Channel scoping restricts the scope of the given name z, reducing its usage only to the process P. For instance, the name z is private or bound to the system $z(x).P$. Channel names which are not bound to any process are said to be free names. This brings us to another channel scoping concept. Since occurrences of the variables in a process will be substituted by the channel names, these variables are bound to the process by the input action. These definitions have consequence on declaration of partner links and also variable definitions into BPEL declaration.

Concurrency concept $P|Q$ means that process P runs independent of process Q. Both processes can communicate channel names between them by performing input/output actions on a common channel. When multiple processes are running concurrently, we have nondeterministic communication.

A conditional statement will check for the equality of the two channels x and y, (x = y). If these channels are found to be equal, then the computation continues as P. Otherwise, if the channels are not equal, the computation continues as Q.

A replicated input action has the same behaviour as an input action. The difference is that upon communication, it produces a copy of itself before proceeding with the execution, thus leaving an intact copy within the system. All these formal constructions are associated to transformation rules to BPEL language where higher operators are defined also.

5.2. Numerical example

As an example, we give a sketch of π-calculus specification which describes the organisation of Choleski computation based on web service invocations. This first example allows defining two kinds of data: a sequence of statements and definition of matrix type. This tiny example represents a sequence where input data is first received, a service, called reduce, is invoked and the result is exported to the caller.

$$CholeskiWS(request, response) = \nu\ result \begin{pmatrix} request(matrix) \\ .reduce\langle matrix, result\rangle \\ .\overline{response}\langle result\rangle \end{pmatrix}$$

Figure 1: Higher order pi calculus specification example

Matrix type can be specified as π-calculus terms if analyst is π-calculus expert, but it is also possible to describe this data type through an XML definition. In that case, the used language is not so formal but as rigorous as possible with XML schema language. It provides a language-independent data types. They are computer representations of well known abstract concepts such as integer and float. On figure 2, we use not only atomic data types but also container like list and bag.

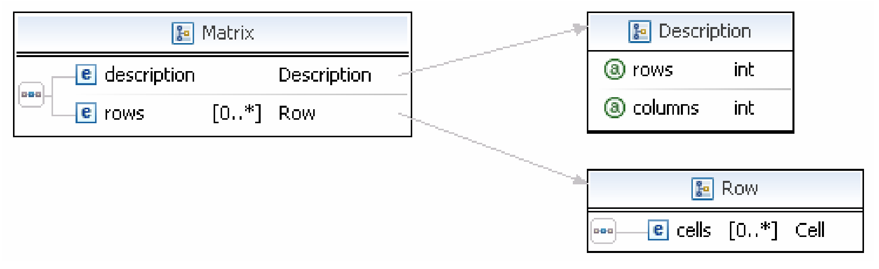

Figure 2: XML schema description of parameter

WSDL plays an important role in the overall Web services architecture since it describes the complete contract for application communication. Each invocation statement is related to a WSDL operation definition. This description can be completed through the experience of analyst. This is about style of invocation, message exchange pattern, parameter encoding, and so on.

A π-calculus analyst can add to his specification (Figure 1) a target namespace for his WSDL definition, just like he would for an XML Schema definition. Anything that he names in the WSDL definition (like a message, port type, binding, etc.) automatically becomes part of the WSDL definition's target namespace defined by the target namespace attribute. Hence, when he references something by name in his WSDL definition, the analyst can remember to use a qualified name through π-calculus specification.

By the end of specification work a set of description is built and it is a basis for control step and generation module.

6. BPEL skeleton and enrichments

Every company has its unique way of defining its business process flow. The use of formal method is another approach for building business process description. The key objective of BPEL is to standardize the format of business process flow definition so companies can work together seamlessly using Web services. Our choice of process algebra allows companies to place formal methods into the business core.

6.1. Analyst role

We have developed a software chain based on users' roles. First, analyst writes his specification through use of graphical editor based on our own framework [15]. Then, he can import schema definition or external π-calculus specifications (Figure 3 *HOpiCalculator*). Of course, this tool checks syntax and dependencies of all descriptions. It provides also metrics about service call hierarchy depth, more complex scope of message type and a consequence about specification cohesion. It is a measure of how strongly related is the processes expressed by the formal specification modules.

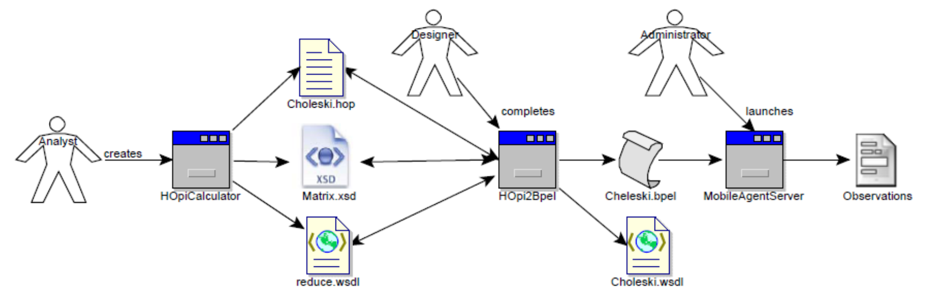

Figure 3: a software pipeline for BPel script creation

6.2. Designer role

Then, designer can refine specification and types by adding details (Figure 3 *Hopi2Bpel*). This one corresponds to a user who knows SOA architecture. While there are many definitions of a service-oriented architecture, there are four particular components that designers agree on. The service consumer is the kernel of his activity. The designer creates a component module that requests services. The service consumer finds the service provider in the service registry and sends a service request and executes the service function. Also designer has a clear overview of the whole architecture of the service graph.

The service provider is how this business process is accessible to other applications or software components. The service provider publishes its contract (*Choleski.wsdl* url) in a service registry as a standard service to make itself discoverable to service consumers. Designers provide values to attributes of binding element. The name attribute, which is chosen by designer, defines the name of the binding, and the type attribute points to the port for the binding.

Designer interacts also to set two attributes: style and transport. The style attribute can be "rpc" or "document". In this case we use document. The transport attribute defines the SOAP protocol to use. In this case we use HTTP protocol. The operation element defines each operation that the port exposes. For each operation the corresponding SOAP action has to be defined. Designer must also specify how the input and output are encoded. In this case we use "literal".

The service contract is a specification of the precise way in which a service consumer can access the functionality of a service provider. The service contract informs the service consumer of the acceptable message format of a service request.

6.3. Administrator role

A service registry is naturally a network accessible registry that accepts and stores service descriptions from providers and makes them discoverable to service consumers. The administrator is end user of BPel scripts. He gives a BPEL description to an evaluator. This one has to have access to a list of service registries. In our context, this is a set of mobile agents belonging to an agent community (Figure 3 *MobileAgentServer*). This is a structure where a set of mobile agents have specific permissions for working.

With these components in place, it is possible for service consumers to easily find services that can provide the functionality they require to complete a task. By simply querying the service registry with a specification they will be provided with a list of any service providers that can offer the function they require. Administrator role is more complex, because process observation is also one of his key activities. New metrics need to be defined. In our context, metric creation means building new communities of mobile agents and deployment over same network. This belongs also to the mission of administrator.

7. BPEL engine through mobile agent community

We adopted a two level architecture [16]. The upper one defines software architecture and how components are defined with stereotype. This description is useful for understanding technical frameworks. The lower level of architecture describes how our components are placed onto the node of network and it gives a map of available services.

7.1. Software architecture

The main part of our platform is called mobile agent server. It first, receives requests from clients and delegates given activity to a mobile agent. At the beginning, a mobile agent is enough for a treatment, but it could need help if the business process is too complex. This means that it is not a simple sequence of web service calls.

BPEL is based on Web services in the sense that each of the business process involved is assumed to be implemented as a Web service. Also, as mentioned before, all elements of a computation have to be equipped for their future use. Also, when a request is received about a given business process, we built a new business space,. We consider a business space as persistent object exchange areas through which remote business processes coordinate actions and exchange data. Such an approach simplifies the design and implementation of sophisticated distributed applications. This distributed structure is useful at runtime, because it contains all parts of the execution context and it manages some aspects such as transaction management, state backup and part of safety control.

We have already defined set of BPEL scripts for some of business processes. They are all about statistics; and more precisely two subsets of statistics descriptive and inferential. They allow us to create graphs and charts, averages, dispersion of data, probability and its distributions etc on large data sets. Our experience about BPEL language allows us to define two different types of business processes: executable processes and abstract processes. Our models are based on actual process of statistics domain. Also, we declared only executable process which can be executed by an orchestration engine at least. As example, we built a BPEL process called "PoissonGuess", which implements Poisson change-point estimations using a BPEL process and three Web services: DataExtractor, PoissonSolver, ResultFormatter.

7.2. BPEL mobile engine

A business processes definition, describes a variety of elements: the partners in the current business process, the messages that need to be transmitted, the type of Web services that are required and the kinds of Web services connections that are required for operations. All these data configure our mobile agent before starting its mission. So, it has to check relations from the BPEL declarations to the WSDL definitions which define the interface details of the Web services. Then, the BPEL sequence defines the behaviour of our mobile agent; this means the web services should be invoked, and they can be invoked synchronously or asynchronously.

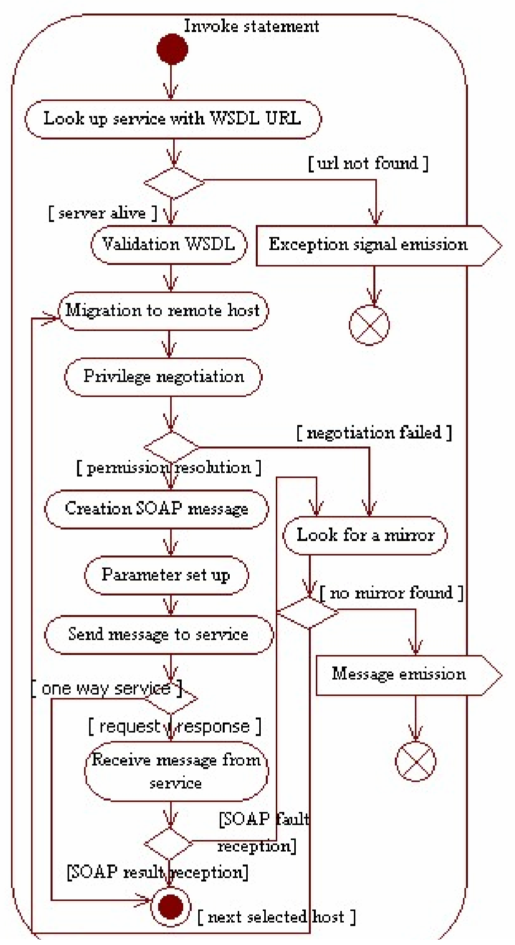

Figure 4: State chart for Invoke command

For all of the local actions, our implementation of BPEL language is quite similar to official reference. But when it's about invoke statement or data collection; we replace a remote call with an agent migration and a local call.

Of course, our agent uses same standard languages (SOAP, WSDL and UDDI), but we reduce message volume and we consider as pre treatments some aspects which were evaluated before during a call. This takes into account a part of security controls, event tracking, etc. As example, the interpretation of invoke element is one of the main changes.

This XML block (Figure 3: script BPel) contains all the details to realize a service call. PartnerLink attribute provides a reference to a partner declaration at the beginning of the BPEL definition. This partner declaration gives how to find WSDL definition of the given service.

Figure 4 describes how an agent interprets invoke statement. It divides this operation into two steps: migration and call. If the url of WSDL definition is not valid, an exception is raised and the business space is notified on this problem, then a message is logged into anomaly report. This will be the trace that a business process failed and then a robot could collect them and a recover strategy could be implemented.

When the url is valid, it asks remote web server for receiving WSDL stream. Then, after its reception, he uses the service tag (from WSDL stream) to find out its destination. It notifies its business space about its future migration and look up the acceptance service of its next destination. If this technical service is available, it will move from its current host to the host where the wished service is published.

For this migration, mobile agent needs a technical service, called acceptance service, which has to be defined on next host and published into local service register. An acceptance service allows knowing if a mobile agent has enough rights to perform all the actions onto the current host. Our objective is to prevent anomaly before raising an exception, when mobile agent does not have enough permission to access to local resources. This agent checking is done by Negotiator component.

Controller exception can not be totally suppressed because, mobile agent can have permission to read a local resource which changes its mission (for instance, a file contains a list of email addresses or url and they will be used to notify that the mission is ended). If this agent doesn't have permission to send message, Supervisor component will raise a controller exception, the business space is notified and the business process will be suspend.

The negotiation between agent host and mobile agents looks like a frontier where all incoming agent are placed into a queue. This is a partial filter against access violation and misleading actions.

7.3. Negotiation at entrance

When an agent host receives a mobile agent, Negotiator role is to check the action list, the mobile agent wants to realize. And then it validates with the permissions, it has for this agent. If these permissions are not sufficient for the evaluation of the agent mission, the negotiation fails (see figure 4) and mobile agent has to find another agent host where the service is declared.

For the permission assignment, mobile agent has to have certificates and details about its origin and its owner. When agent host accepts this new worker, it is launched in parallel with Supervisor component which observes what mobile agent will really do during its mission. This is particularly crucial when mobile agent sends requests about importation of other agents or when it uses value of local resource.

When negotiation succeeded, agent can realized its service invocation. Because of interoperability, this call is done through an XML message. Our current prototype uses SOAP message which is built by XSL transformation from the WSDL definition. If it is a one way service, no response is expected and the invoke statement is done. If the message exchange pattern is a request-response invocation, it will be blocked until the reception of the SOAP result. If a SOAP fault is received, then mobile agent has to look for another server where the service is available. When a result is received, mobile agent can interpret next BPEL instruction. This on is depicted by another state chart, maybe another web service invocation or an assignment or another BPEL activity. But this next step can be done on the same agent host (under the control of the same Supervisor component) or not depending on kind of instruction (local service or not).

Main events are recorded into local registers: agent importation and exportation, but also negotiation success and failure, local resource access and extra data about current business process. This is used to trace behaviour and also to look for a better execution time. Blockings occurs when resources are not shared, and then object locks are declared. These enable multiple agents to independently work on shared data without interfering with each other. The mutual exclusion refers to the mutually exclusive execution of monitor regions by multiple mobile agents. At any one time, only one agent can be executing a monitor region of a particular monitor. If two mobile agents are not working with any common data or resource, they usually can't interfere with each other and needn't execute in a mutually exclusive way.

Agent host can apply an order among all current mobile agents. A higher priority agent that is never blocked will interfere with any lower priority agent, even if none of the agents share data. The higher priority agent will monopolize the CPU at the expense of the lower priority agents. Lower priority agents will never get any CPU time. In such a case, a monitor that protects no data may be used by agent host to orchestrate these agents to ensure all agents get some CPU time. The monitor strategy is based on quota distribution which depends on role of mobile agents on current agent host. So, a host can privilege mobile agents with a Statistics role compared to others with role Collector. Similarly, a host can fail negotiation because it guesses; there have already been enough agents with a given role. Idem, in case the negotiation fails, mobile agent is looking for a mirror site (see figure 4).

8. Conclusion

Through this work, we have explained our approach of business process definition and also its evaluation by the use of mobile agents. The orchestration engine has a new strategy to move decision power near to the location where web services are called. This way of evaluating BPel changes error recovery technique. This emphasizes the importance of access to resources compared to the knowledge of service in a directory. Our next direction is about negotiation control and adding new details into WSDL definition of web services. This will allows designer to place into his BPEL definition, more precise information about migration preparation of mobile agents.

References

[1] Service-oriented computing : (ICSOC 2005) (Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005) (proceedings) International conference on service-oriented computing No3, Amsterdam , PAYS-BAS (12/12/2005) 2005 , vol. 3826, pp. 490-494[Note(s) : XVIII-597 p.,] [Document : 5 p.] (5 ref.) ISBN 3-540-30817-2;

[2] Kadima H., Monfort V. Les Web Services, Techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML. Dunod. Paris 2003.

[3] Peltz, Chris. Web Services Orchestration: a review of emerging technologies, tools, and Standards. Hewlett Packard, Co. Janvier 2003.

[4] Sanlaville, Sonia. Environnement de procédé extensible pour l'orchestration Application aux services web. Thèse de doctorat, Université Joseph Fourier, Décembre 2005.

[5] Antonella Chirichiello and Gwen Salan. Encoding abstract descriptions into executable web services: Towards a formal development. In 2005 IEEE/WIC/ACM International Conference on Web Intelligence WI'2005, Compiegne, France, September 2005.

[6] A. Ferrara. Web services: a process algebra approach,. In Proceedings of the 2nd international conference on Service oriented computing, pages 242{251, New York, NY, USA, 2004.

[7] Nick Russell and Wil M.P. van der Aalst, "Evaluation of the BPEL4People and WS-HumanTask Extensions to WS-BPEL 2.0 using the Work□ow Resource Patterns", Department of Technology Management, Eindhoven University of Technology, GPO Box 513, NL5600 MB Eindhoven, The Netherlands

[8] Assaf Arkin, "Business Process Modeling Language (BPML)." "Proposed Draft" [0.4] specification from BPMI.org. Draft 0.4, 3/8/2001. 155 pages. edited by Ashish Agrawal (Intalio,Inc.). Available 2001-03-08 from BPMI.org.

[9] W3C. Web Service Choreography Interface (WSCI) 1.0. Accessed November 2002 from www.w3.org/TR/wsci/, 2002.

[10] Web-Services Conversation Language WSCL 1.0, May 2001, http://www.hp.com/go/espeak

[11] Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guizar, A., Kartha, N., Liu, C.K., Khalaf, R., K□onig, D., arin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language Version 2.0. Technical report, ASIS Web Services Business Process Execution Language (WS-BPEL) TC (April 2007)

[12] Bartoli A., Jiminez-Peris R., Kemme B., Pautasso C., Patarin S., Wheater S., Woodman S., « The ADAPT Framework for Adaptable and Composable Web Services », IEEE Distributed Systems Online, September, 2005.

[13] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. Inf. Comput., 100(1):1–40, 41–77, 1992.

[14] Sangiorgi, Davide; Walker, David (2001). The Pi-calculus: A Theory of Mobile Processes. Cambridge, UK: Cambridge University Press. ISBN 0-521-78177-9.

[15] Andreea Barbu et Fabrice Mourlin, "From pi-Calculus Specification to a Simulation of a Mobile Agent Using JINI", in ESM 2004, 18th European Simulation Multiconference, Magdeburg, Germany, June, 2004,

[16] Mâamoun Bernichi, Fabrice Mourlin, "Two Level Specification for Mobile Agent Application," icons, pp.54-59, 2010 Fifth International Conference on Systems, 2010