Deliverable 1:
SOA And Web services for negotiation e-market place: motivation and state of the art

Abstract.
The deliverable1 concerns the WT1. We present our approach to implement a negotiation e-market using an SOA architecture and the Web services paradigm. The proposed approach considers an open environment. In order to avoid the emergence of *ah hoc* negotiation markets, we propose a functional model that supports mechanisms (negotiation protocols) publication. Markets are then allowed to emerge according to their validity with respect to a negotiation mechanism. We use the standard proposition WS-CDL in order to specify the negotiation mechanism as a set of peer to peer interaction roles (Web Services). A minimal set of constraints (using Abstract BPEL) are then derived for each role. Actors desiring to play a role must verify their validity according to the mechanism.

Title :	**SOA And Web services for negotiation e-market place : motivation and state of the art**
Deliverable Id.:	**1.0**
Date :	**30/11/2007**
Classification:	Public/Private
Originating Partner :	**LIP6**
Author(s) :	**LIP 6 & IBISC**
Relevant Task :	Task1.1…
Release Status :	Draft V0…
Project Co-ordinator:	Lip6
Partners:	IBISC

Table of Contents

1. Recall of the CRE context and goals:.... 3
1.1. CRE Context : the Grid4All project 3
1.2. Challenges and goals of the CRE 3
1.2.1. Challenges.... 3
1.3 General idea of the solution.... 4
2 Two negotiation protocols :.... 6
2.1 Single Shot Protocol.... 6
2.2 Iterative Protocol.... 6
3 SOA as an architecture for negotiation e-market.... 7
3.1 SOA.... 7
3.2 Functional Model of negotiation e-market.... 8
4 Web services FrameWork.... 11
4.1 BPEL and Abstract BPEL.... 11
4.2 A choreography of Web services the WS-CDL language.... 12
4.3 From WS-CDL to Abstract BPEL : the e-market framework.... 19
5 About the relevance of the proposition to the grid4all project.... 22

1.Recall of the CRE context and goals:

1.1. CRE Context : the Grid4All project

The increasing interconnection between computers has created the vision of a computational and more generally a services Grid. Within the Grid, the resources – computational, storage, and also applications are rendered available to anyone participating in the Grid. In scientific environments this is often referred to as a resource sharing model, in which organizations can take part if they share themselves their idle resources.

Grid4All is an IST project started in 2006 under the Call 5 of FP6. In this project, we have taken a market based approach where participants trade on computational resources and all sort of IT resources – storage, data, and applications.

Grid4All is a Grid targeting users on the Internet – and not just confined to a restricted set of organizations. Users on the Internet possess IT resources which they can trade to other users to their own personal profit -- allocation of goods are based on the supply and demand. Current markets rely on a centralized client-server architecture that implements an allocation algorithm to decide which seller transacts which resource and its quantity to which buyer. As opposed to this, ephemeral markets are markets that arise and fade spontaneously according to needs and use without needing a distinguished persistent agent.

Grid4All aims at proposing a virtual e-market place which promotes on the one hand the dynamic and spontaneous creation of markets on need and by those who need them – either suppliers or consumers. On the other hand, the e-market must able to control the validity of the created market according to the set of specified negotiation mechanisms such as English auction, iterative auction, etc.

1.2. Challenges and goals of the CRE

1.2.1. Challenges

Many negotiation based electronic markets have been tackled for closed distributed systems such as Tycoon [TYC05] Sharp [SHR03] These systems propose a single scenario of trade using one and unique negotiation protocol that is deemed best suited to the traded resources, the context and environment within which the resources are used. Dealing with an open market, as claimed by the Grid4All, is a new topic and raises a set of challenges. The challenges can be summarized in the two following points:

An open environment The virtual e-market will be realized in open environments, here the Internet. In an open World, functional and semantic heterogeneity persist. It is unrealistic to suppose that market actors (resource, provider and consumer) define from scratch (while instantiation of a protocol role) their behaviours each time they are involved in a negotiation. It is also unrealistic to restrict actors to implement a specific negotiation protocol. Being developed separately, there is no reason to assume that any two given actors use the same (i) negotiation protocol (ii) the same design of a given protocol (iii) abstraction level while implementing roles – and especially for the consumer agents as it is summarized in the figure below.

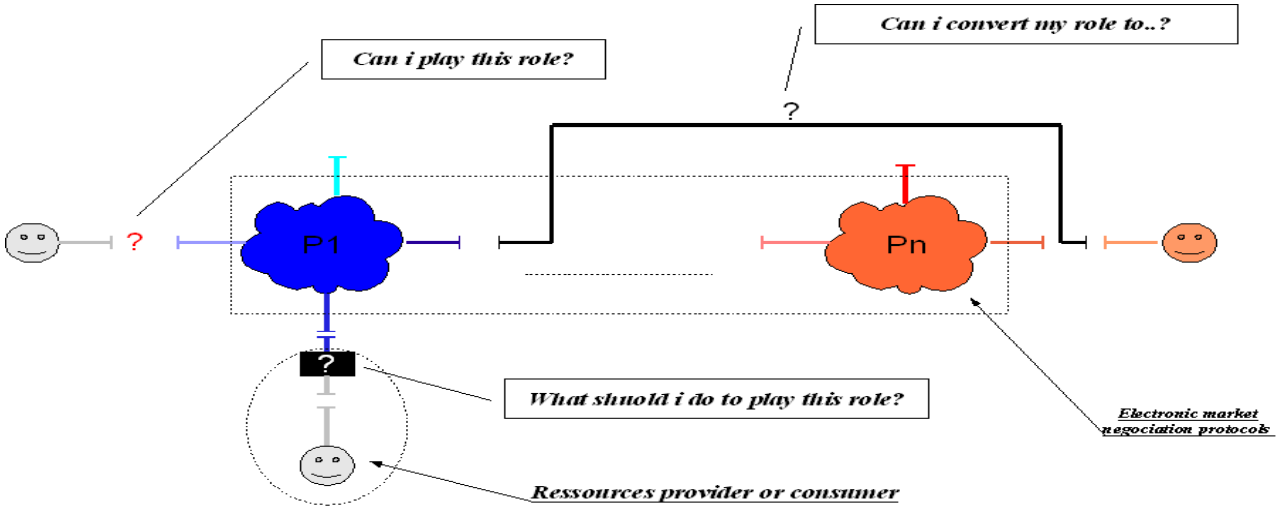

Figure 1: Heterogeneity problems arising within an open negotiation market

Market creation must be regulated according to a predefined negotiation mechanism One of the risks with such open virtual e-markets is that spontaneous markets can emerge and the goods may be traded according to an *ad hoc* process (or in a non-efficient way, bad participant can influence the negotiation process). Since negotiation is a complex process, the emerged market must respect a set of constraints. To guarantee such validity, the market must be able to support negotiation protocol specification and validation procedure. The way of specifying constraints and the way of controlling their respect by the the market agent must be compatible with the open environment, which means that it shouldn't limit the openness of the environment.

1.2.2 Goals

The scope of this CRE proposition is to provide an architectural and technical (framework) solution to realize the pre-described market and face the two cited challenges; the architecture proposition must be soluble in the target environment (here the Internet) to deal with the heterogeneity problems. In addition, the solution must give a functional model and methods that lead to only valid markets to appear in the virtual e-market. The solution must fit this requirement:

- The solution must enable the creation of new negotiation mechanism (protocols).
- The design of a new market mechanism must be easy, rapid an enough expressive to describe the constraints required by a mechanism
- The solution must propose a functional model that guarantees on the fly valid market establishing.

1.3 General idea of the solution

Recently, Service Oriented Architecture (SOA) and its implementation using Web services has shown a strong efficiency in dealing with the heterogeneity of a set of interacting peers on the Internet. One of the main application domains of the SOA and the Web services framework is the inter-organizational business process: E-business. The main idea of the proposition is to consider a negotiation as an inter-organizational process where each of the participants executes internal and external 'business' processes to reach a business objective (price and winner determination). The SOA approach is valid within the e-market place -- market services may be decomposed into elementary services at a lower level to foster a wide and large number of implementations. The Web Services and WS-based process

definition standards provide mechanisms for defining negotiation processes that can be understood and deployed in a platform-independent manner.

The sequel of this deliverable is organized as follows: **section 1** presents two examples of negotiation protocols proposed by FT partner to validate our approach. **section 2** explains our architectural choice, the SOA. **section 3** details the architectural adaptation and the functional model of our proposition to deal with mechanisms and market instances. The **section 4** is dedicated to the Web services framework and languages and their relation to the proposed functional model. Finally, **section 5** discusses our solution and its relevance to the project.

2 Two negotiation protocols :

In order to validate our approach we asked FT (Grid4All) for representative examples of negotiation protocols that may be used in the Grid4All project. The result was two UML like negotiation mechanisms. The first mechanism is the single shot (was produced conjointly in order to fix terminologies) and the second is an iterative shot that was fully produced by FT and validated by Lip6 partner. The FT input protocols are described in the following. These protocols will be used as illustrative example of the proposed methods.

2.1 Single Shot Protocol

In the single shot protocol, sellers and buyers seek to find a market respectively to sell and buy a good. Initially, sellers and buyers have to register before creating a market or getting a market reference that will be used to send bids. Each participant registers by sending her information to a semantic information service (SIS).
Once registered, participants can choose to join an existing market or to create one if no market is found.
An auctioneer is then in charge of receiving participants' bids. Each participant can bid only if she previously registered and can only bid once in a given market.
Participants can register to an existing market at any time until a "stop-registration" time, and bids can be received until a "stop-bidding" time. Both timeouts are predefined on market creation by the initiator (which is in charge of configuring the market).
When the "stop-bidding" timeout is over, the winning participant and price are determined by the auctioneer. Finally, an agreement manager builds contracts between the winning participant and the initiator (I.e. a buyer and a seller.

2.2 Iterative Protocol

In the iterative protocol, sellers and buyers seek to find a market respectively to sell and buy multiple items, i.e. more than one type of good. Each market adresses a set of items with a given quantity for each item, which does not evolve during the course of the auction.
As in the single shot protocol, an initiator role configures the market (timeouts items types and quantity). Participants have to register before getting a market reference, if any, and to send bundle bids
on the objects traded at the auction. Contrary to the single-shot protocol, bids may be withdrawn.
The auctioneer is then in charge of matching participants bids against the initiator offer. In this protocol, the auction can iterate if the matching was not successful.
When the winning participant and price are determined by the auctioneer, an agreement manager builds contracts between winning buyers and sellers.

3 SOA as an architecture for negotiation e-market

3.1 SOA

Service Oriented Architectures (SOA) [SOA03] have introduced a new organization of software, based on services: self describing and loosely coupled interacting software components that support the rapid and low-cost composition of distributed applications. Services constitute the next major step in distributed computing, because they provide a uniform view of heterogeneous software entities that populate open environments (e.g., the Web or pervasive computing environments) and also make a separation between service interfaces and services implementation (whether they are logical services or physical ones). In this area, Web services, one of the SOA implementations, is becoming an incontrovertible paradigm for the development of applications in open, large-scale, distributed environments. One of the most important issues in SOA is the automation of service composition, either to ensure a given user task or to make services collaborate altogether to build added-value composite services.

The advantages of SOA are many-fold but one of the most relevant advantages of using SOA as architecture base for Grid4All negotiation e-market is the capability of the service model to deal with heterogeneity either on the nature of the services (application or physical resources) or on the format (access interfaces and description languages). This offers a robust and inter-operable environment for providing and consuming resources in an open world where the only assumption done on the actors is that they act according to a SOA model. The SOA model defines three actors:

- **service provider:** an agent providing a service
- **service consumer**: the agent that consumes a service (client)
- **intermediaries** : entities that connect the requirements of the consumer to the providers services.

The **fig 2** represents the functional model of a SOA. First, a service is created, then it is published in the intermediary in a readable format. The consumer locates its need in the intermediary and then consumes the service.

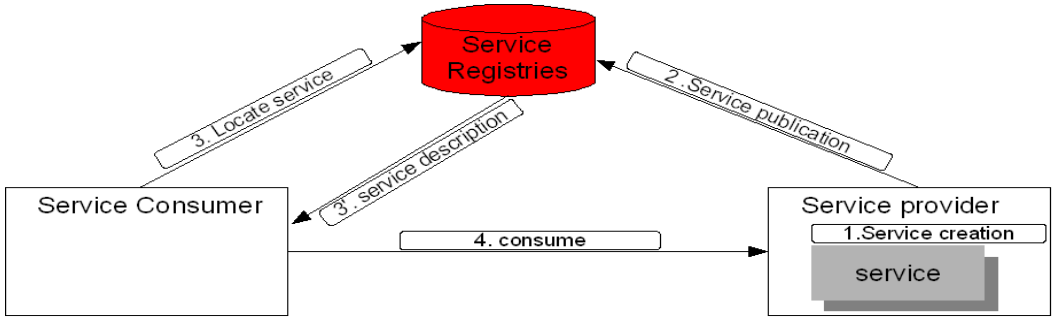

Figure 2: SOA functional model

The SOA approach is a valid architectural choice for the e-market architecture -- Market services also may be decomposed into elementary services at a lower level to foster a wide and large number of implementations. In addition, most of the Grid Platforms (eg. Globus [GLO02]) are switching to an SOA architecture for the resource consuming. The use of SOA as an architecture for the Grid negotiation e-market adds uniformity in the whole process : negotiation for a resource and then resource consuming.

3.2 Functional Model of negotiation e-market

The functional model of the present proposition can be summarized in the **fig 3**. The corner stone of our functional model is that it is working as a SOA functional model (**fig 2**): creation, publication (in public registries), discovers and plays (consume the service). The specificity of our solution, directed by the specificity of the CRE problem, is to apply such functional model at two different levels of abstraction ; the first level, called **class level** or **mechanism level** concerns the negotiation mechanism. The second, called **instance level or market level**, concerns a specific market that instantiates a specific mechanism. The functional model propose also a link between the different level to allow only valid market according a mechanism to appears.

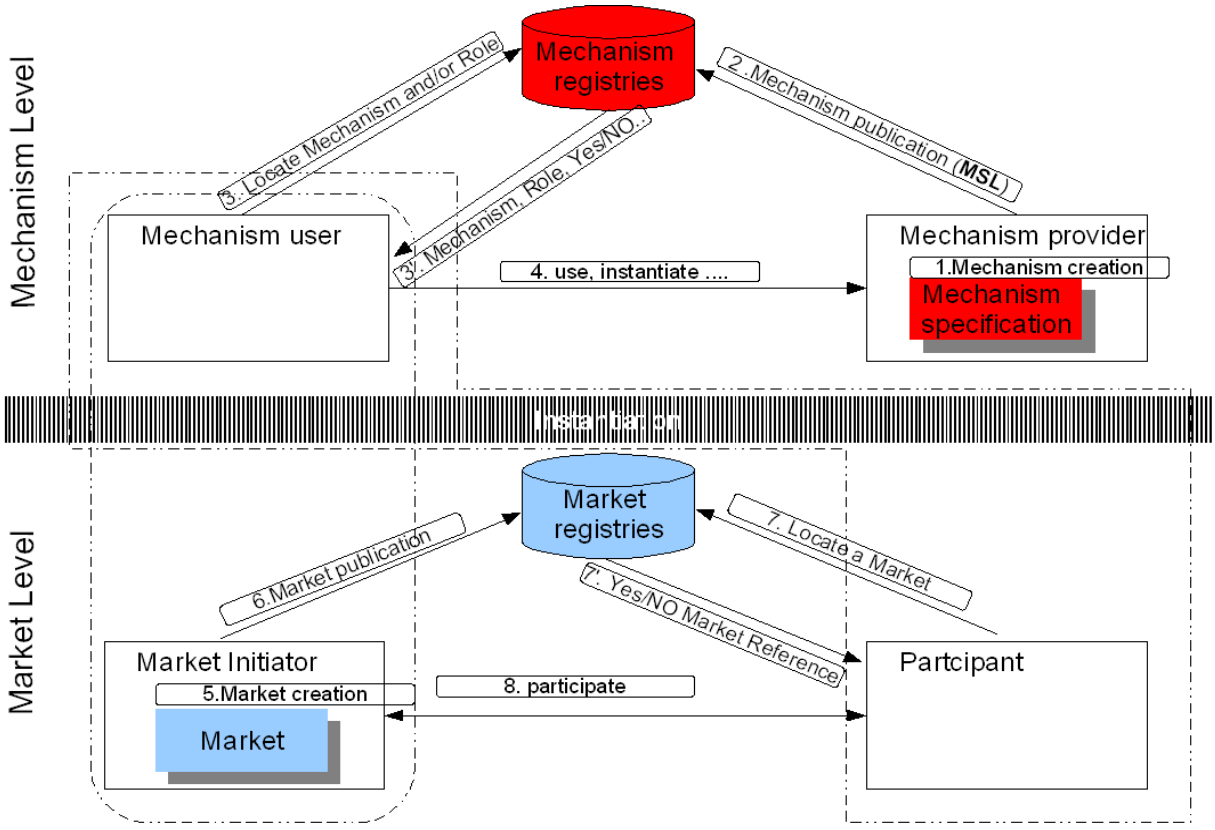

Figure 3: functional model of the Grid4All e-market : mechanism vs market

3.2.1 Actors

In our functional model, we distinguish two classes of actors: the class level actors and instance level actors. Some of them can define the same agent.

A) The class Level actors

- **Mechanism provider** : represents the agent that should provide the market by mechanism. It can be the market administrator or a third party. We can imagine that the market supports that actor can propose mechanisms.
- **Mechanism registries** : the mechanism registry represents an intermediary agent that allows to market initiator and participant to discover existing negotiation protocols
- **Mechanism users :** are agents that use a mechanism to instantiate a market or to instantiate a role in a market according to a mechanism. Mechanism users are the market initiator and the market participant of the instance level.

B) The market Level actors

- **Market initiator** : is an agent that initiates a market for a specific good and according to a specific mechanism. See the model description;
- **Market registries** : they allow the market initiator to publish its market and to participant to discover the market.
- **Market participants** : are participants of the negotiation for a specific market.

3.2.2 The model

The functional model describes the whole picture from the creation of a mechanism until a market execution.

A) The Mechanism Level :

The class level aims to define the set of accepted mechanisms in the e-market place.

i. Creation:

A negotiation mechanism (protocol) is added to the e-market by the e-market supervisor or administrator. He/she defines the set of supported negotiation schemas by the virtual e-market. The specification of the negotiation mechanism consists in defining the set of roles involved in the negotiation and their accepted behaviours and the rules (accepted interaction, time, and data constraints) to play such protocol. The mechanism specification is described in machine understandable language (that will be specified in the next section when we detail the framework) for the moment, we call such language Mechanism Specification Language (**MSL)**. From an **MSL** specification we derive for each role its behaviour specification using a Role Specification Language (**RSL).**

ii. Publication :

Two kind of information describe a mechanism: functional and semantic information. The functional information corresponds to the information that explains how the mechanism works. Functional information are composed by the **MSL** specification and for each role involved its **RSL** specification. The semantic information correspond to a set of meta-data concerning the use of such mechanism e.g. for which type of goods the mechanism can be suitable (quantitative, qualitative, one or more goods, etc.). ***We note here that this project focuses only on functional features.***

iii. Discover (step 3 and 3') :

In an open market, the different actors that will participate to an instance of market are unknown in advance and may have their predefined behaviours (it is difficult to suppose that they are created from scratch for the negotiation purpose). The registries offer a set of semantics and functional discovery services in order to help actors to create valid markets. At the semantic level, it can help actors to choose the right mechanism (it's out of the scope of the project), *what mechanisms are allowed for multiple quantitative goods market?* At the functional level, it gives information on a specific role (e.g. constraints), *how a bider must behave in a single shot protocol?* Other services, that will be exposed in the next step of the project (**deliverable 3)** such as the compatibility checking. "*Here is my behaviour, can I be a bidder in an iterative auction market instance?*".
The **RSL** can be discovered in order to give a starting point (specification) to develop from scratch a dedicated actors.

iv. Instantiation:

When an entity identifies the appropriate mechanism and checks that it can play a desired role, it can then participate to a market instance. Here, we distinguish between a market initiator and participant (see market level), both of them can instantiate a role for the market level. The market initiator can instantiate the corresponding market by creating an instance of the market. The resulted instance of the market will then be published in registries of the Market level.

C) The Market Level

The Market level corresponds to the market instances. The creation of a market corresponds to a given mechanism decided by the market creator (initiator). An instance of a mechanism is a set of implementations of each role composing the mechanism. Note here that many implementations can exist, appear and disappear in the market place (dynamic environment).

i. Instance of market creation:

The initiator can, by using the MR, decide of the best mechanism and its capability to be a valid implementation of the initiator role. Once it is done, each execution of that component concerning a specific good corresponds to a market. e.g; if the component was validated as a valid initiator role of the ***single shot*** and if it is executed for a specific good like *CPU time* then that executed instance corresponds to CPU market according to the ***single shot*** mechanism. Note that the initiator and the mechanism user can be the same agent (the dashed rectangles in the figure 2) viewed in different levels.

i. Publication of market instance

The Grid4All e-market maintains also an instance registry IR. The instance registry offers a set of services (those services are defined in the two protocols **section 2** by SIS partner) that allows market instance registration and discovery. The IR describes properties of the market properties: types of goods, dates, constraints, etc. We also find a reference to the type of market (the mechanism) and the agent implementing the initiator.

ii. Discover of market instance (step 7 and 7')

The market is discovered using the IR informations (the participant in the figure). An agent can check for a market of a given good that instantiates a given mechanism. E.g., an agent that checks if its specification is a valid implementation of the auctioneer role of a single shot can participate as an auctioneer in a *CPU time* market instance. Note here also that the participant and the mechanism user (in the mechanism level) can be the same agent (the dashed rectangles in the figure 2) viewed in different levels.

iii. Executing a market instance

Once the created market published and then discovered, the different agent execution instances correspond to the market instance execution.

4 Web services FrameWork

Web services are technologies that aim to implement the SOA on the Internet. The actual web services technological framework is composed by a set of XML-based standards specification that insure interoperability in different levels; (i) communication interoperability (XML, SOAP, conventions etc.), (ii) functional by a set of interface description languages using WSDL for simple services and Abstract BPEL (or WSCI) for stateless Web service and (iii) semantic interoperability using semantic Web languages OWL, RDF or more specific OWL-S and also non functional such as security Qos, etc.

WSDL (Web Service Description Language) [SDL07] is responsible for formalizing the service features according to a schema that is very similar to a typical API definition. WSDL is an XML based language able to specify the service feature we have just described in text form. The first element comprising a WSDL specification is the service, which identifies a set of services, each specified by a port. It should be noted that a port only represents the physical address where the service operates and the protocols the user should adopt to communicate with it, with no description of the functionalities provided. This aspect is defined by the ***portType***, directly associated with the port, which is responsible for defining the available operations. Hence, ***portType*** defines what the service does, whereas port defines where it is.

The WSDL model is restricted to only very simple computation services (it presents services as a set of independent operations). Hence one of the main issues of SOA is services reuse and composition on services, this leads to more complex services that need a long running interaction (interaction protocols), working either in central configuration (coordinator) or in totally peer to peer configuration. To this aim, two extensions of Web services technologies are currently investigated: the *orchestration* and the *choreography*. The first aims to combine existent Web services by adding a central coordinator (orchestrator) which is responsible of invoking basic Web services, according to a set of control flow patterns. The second is referred to as a Web service choreography, which does not assume the existence of a central coordinator but defines a conversation that should be explicitly considered by each participant Web service (called partner). In most cases, choreography and orchestration are used in complementarity.

WS-CDL and BPEL are not to different approach to realize different configurations of services composition but different levels to consider the same thing. In the remainder of this section we will present a brief introduction to these two languages and explain how they will be used in the project (integration of the framework on the proposed functional module).

4.1 BPEL and Abstract BPEL

BPEL4WS (BPEL for short) [BPE07] is a joint proposition by IBM, Microsoft and BEA in 2003. BPEL is an XML-based specification that offers a grammar for centric-oriented Web services composition languages. The centric aspect is due to an orchestration approach; The BPEL languages aim to create composite Web services by combining existent ones. For that, the language is defined by two types of activities: the basic one and the structured one. The former aims to define the services in term of (i) its communication with the external environment; what service needs to be invoked and what are its outputs (***receive*** and ***reply*** activities) (ii) the used Web service to create the composite one (the ***invoke*** activity) (iii) data manipulation and time activities (e.g. ***assign***, ***wait***, etc.). While the later, the structured one, aims to define the program logic to structure the different basic ones and thus in term of workflow like constructors (***sequence***, ***choice***, ***while***, ***flow***, etc.). In order to explain the centric composition approach used by BPEL (orchestration) we consider the following example. Two Web services exist: one belongs to an airline company and offers a service of consulting flights and the second belongs to a hotel company and offers a service of booking rooms. A travel is composed by a flight and an accommodation. A trave agency can create a third service that uses the two previous

ones to offer travel services. The resulted service will ***receive*** a travel request then invoke the flight service to get a list of flights (with respect to the requested dates constraint) and then (sequentially) invoke the hotel service, then after comparing the different combination of prices ***replies*** to the requester by sending the set of travel plans. This example shows many aspects:

- **The composition :** by creating a service based on other services.
- **The coordination:** this is done by structuring an order of the used Web services: the consulting service before booking and then the paying services, etc.
- **The interaction protocol:** by projecting the BPEL code on the communication activities (**receive** and **reply),** the different structured and time constraint we obtain an interaction protocol that can be used by other web services to invoke it correctly (see **fig 4**).

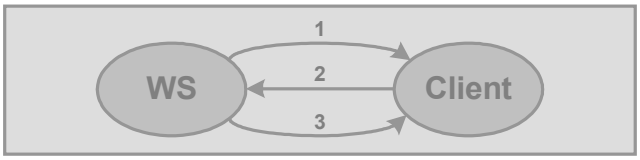

Figure 4: orchestrated Web services may needs a long-runnning interaction protocol

The last point is materialised by what we call abstract BPEL. Abstract BPEL expresses an interface type (or interaction contract) that must be respected for a correct use of the composite service. Now if the services change its implementation and do not invoke only one airline company but ten, the abstract BPEL remind unchanged because the modification does not affect its interaction protocol of the service. The abstract BPEL represents a class of BPEL services which interaction protocols respect the same contract. i.e an abstract BPEL can have multiple implementations and whatever the implementation is, the use of the services remind the same. Abstract BPEL can be used as a contract to guarantee that a component, here a service, behaves in a certain way and respect behavioural and time constraints.

In an open environment, different realisations of a service interface can be allowed and their validation pass throw their respect to a given abstract BPEL contract. This fit exactly our aim of the virtual open negotiation market. We allow actors of the same role to be different but for a correct negotiation process. The implementation must respect a set of behavioural constraints that can be expressed by an Abstract BPEL specification. We note here that Abstract BPEL is a machine understandable language and also the derivation of an abstract BPEL from the BPEL code is automatically possible[HAD05] so the process of comparing a BPEL service to an Abstract BPEL can be realised automatically. This is what is claimed in the functional model presented in **section 2**. The comparison relation between a role specification and an actor behaviours that desire to play such a role will be addressed in WT3 by using a compatibility relation.

4.2 A choreography of Web services the WS-CDL language

Web services choreography concerns the interactions of services with their users. Any user of a Web service, automated or not, is a client of that service. In the case of orchestrated Web services (previous) the composite Web service plays a client role for all its invoked Web services and a services role for its client. It can happen that one of the service client is one of its providers, we call such a case a peer to peer collaboration between the services (and not through a coordinator as a central approach) and the resulted services are choreographed Services. A choreography is a composition of services where there is no central service that orchestrates the collaboration protocol but the collaboration is realized point to point (distributed configuration), as shown on **fig 5**.

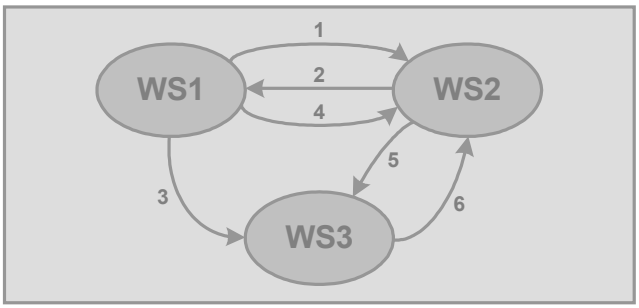

Figure 5: peer to peer Web services composition

To understand such a situation, we must adopt two point of views. From each service peer (or partner or participant) point of view, it will behave as an orchestrated web service, it will invoke and be invoked according to a specific order (local collaboration protocol). From the global point of view, the composition of the local collaboration protocol must fit the expected service collaboration (global collaboration protocol). **Fig 6** shows two possible approaches to design such distributed composition:

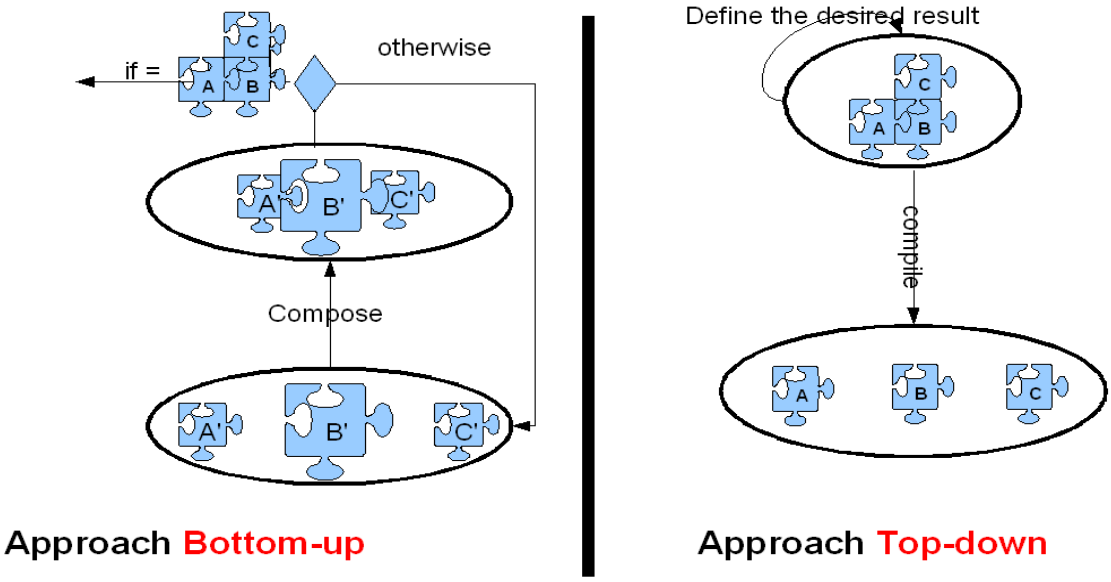

Figure 6: Two approaches for peer to peer Web service composition

(i) bottom-up approach defines each peer interface (the interface contains services interaction primitives and also the partner operation invocation) independently and then composes them and check whether they respect the global behaviour. The process is iterated until success or (ii) top-down approach, which is very used in cryptographic protocol design where the wanted global behaviour is designed in terms of order and time constraints on interaction between partners and then derives each partner interface by projection mechanism (or compilation). WS-CDL (Web service Choreography Description Language) is an XML language that aims to realise the second approach.

WS-CDL is an XML-based language that describes peer-to-peer collaborations of parties by defining, from a global point of view, their common and complementary observable behaviours, where message exchange occurs (order), when the jointly agreed ordering rules are satisfied. WS-CDL propose the way to describe the services peer collaboration without adopting a particular partner or peer point of view . In the following we describe the main concepts that compose a WS-CDL specification.

A WS-CDL specification [CDL07] is defined by a set of declarations that outline the choreography. A choreography specification is defined by a package. A package is composed by a set data channel types declaration, roles, participant and global protocol interaction specification. The **fig 7** represents the structure.

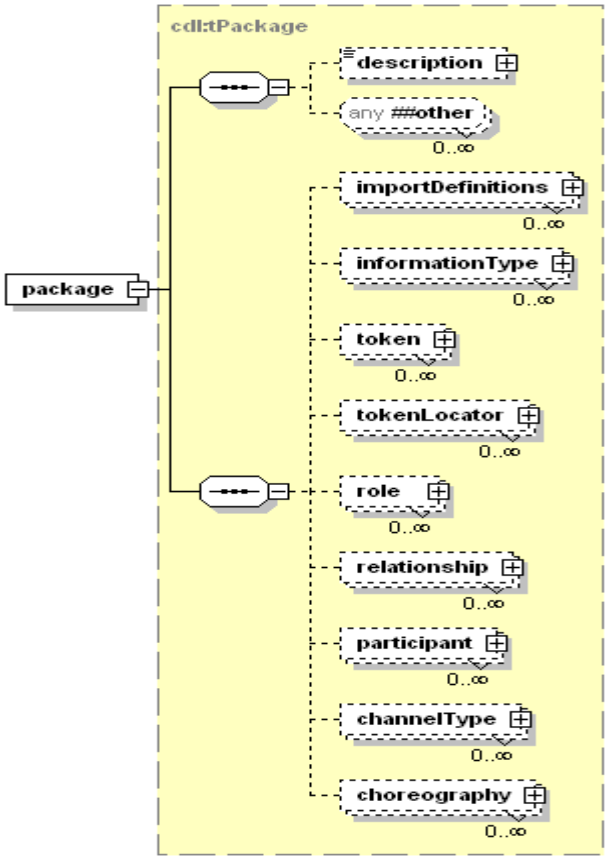

Figure 7: the WS-CDL file structure

4.2.1 Participant Roles and Relationship

A choreography is between a set of ***participants***. The participant here is an abstract entity that corresponds or can be instantiated during the realization by a specific partner. Participants interact with each other in order to collaborate. Their collaboration is realized by a set of interactions. Their interaction can be defined as a set of roles. Each participant can play one or more roles in a choreography (e.g. in the single shot protocol, we can identify two roles associated to the SIS participant partners, SIS participant, market register roles and market finder). Roles of the same participant are at the most cases independent. So a role identifies a unit of observable behaviour that participant will exhibit to interaction with other participant (more precisely other participant roles). Two roles that may interact (e.g. bider role and SIS finder role) must be related by a an oriented ***relationship.***

In this part of WS-CDL specification we answer the question *who?* by defining the structure of the collaboration ; *who will be implied? who will collaborate with whom?,* etc. Figure 8 shows clearly this point.

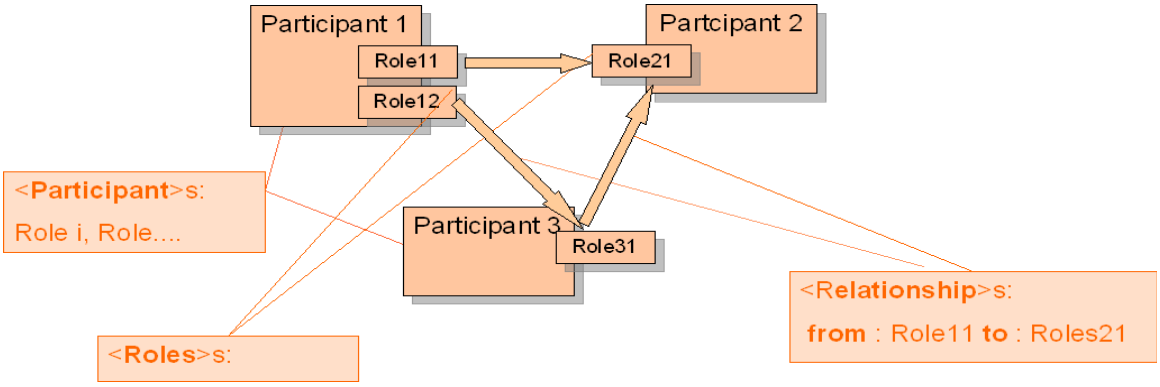

Figure 8: The Participant, Role and Relationship of the WS-CDL specification

4.2.2 The channels

The channel is one of the important aspects of WS-CDL specification. It represents a collaboration point of a role. A channel is identified to the role it belongs to. A channel defines an interaction model, request-response, solicitation or notification, accepted by the role. A channel can be defined exclusively as a data exchange channel of a role or as a channel that can potentially exchange data of other channel. The channel exchange is one of the strong point of WS-CDL it allows to discover in a dynamic way (during runtime) other partner references. A simple example is choreography between a buyer, a seller and a shipment service. During the first part of the choreography the interaction is essentially between the buyer and the seller in order to make an agreement on the price. Then the seller contacts the shipment service and sends him the buyer channel in order to complete the interaction and make an agreement on shipping modalities. In this case, the shipment service discovers the buyer partner during runtime. This issue of WS-CDL (inherited from the Pi-calculus model) leads to more dynamic collaboration schemas. In the Grid4All project the actor initiator (which offers the good) will be discovered by the winner during runtime (We will show in the **deliverable 2** how the feature of WS-CDL is useful to make dynamic markets) in order to obtain the good. Actually, while our market actor is realized as Web service, a channel is the data that my contain information on the Web service reference URL address and possibly a set of correlation data in order to identify a specific instance of the target partner.

Channel
- **name**
- **action** = request | respond | request-respond
- **passing** = list of passing channels
- **role** = target of the channel
- **reference** = variable storing the reference to the participant
- **identity** = unique ID descriptors – correlation

Figure 9: The WS-CDL channel type Element

Channels declaration are types of channel definition and their use in choreography is assimilated to a special kind of data. Channels instances are manipulated as variables (see next section) of the target channel types. Each channel type must indicate the role that it belongs to. If the channel can

be used to pass other channel data this must specified as ***passing*** element by indicating the types of the channel instances that can be passed throw.

4.2.3 Data oriented collaboration

A WS-CDL document allows defining information within a choreography that can influence the observable behaviour of collaborating participants.

A) Data types

Information types describe the type of information used within a choreography. By introducing this abstraction, a choreography can use the data types schemas shared by partners. Data types schemas are imported in the choreography. This issue is important when a set of choreography can share the same data types system. In the case of Grid4All, we can imagine the use of shared data types system that can be used by the market administrator and also the market actor to define or to map their data type systems to.

```
<informationType
 name=
 type="qname"? |
  element="qname"? />
```

Figure 10: The WS-CDL informationType element

B) variables

Variables capture information that define the data manipulated within a choreography:

- *Information Exchange Variables* they are application oriented data eg. the bid values or the good description etc.. Within a choreography such variables are used to:
- Populate the content of a message to be sent, or
- Populated as a result of a message received
- *State Variables* they contain observable information about the state of a role as a result of information exchanged. For example when the market send the “begin of session” message that contains the current price to the bider, the bider participant could have a *State Variable* called "currentPrice" and sets its value when receives the message.
- *Channel Variables* that contain channel instances. For example, a channel variable can contain information such as the URL to which the message should be sent. Those variables can be used as message variable when the information about channels are exchanged

The value of variables:

- Is available to all the roles by initializing them prior to the start of a choreography (belongs to more that one role). The channel variable value of a given partner is known by all the roles that invoke him without previous channel value exchange (the first exchange). In the case of Single Shot protocol the channel of the SIS (find-market) role is known by the participant.
- Can be made available to a role by populating them as a result of an interaction (communicate values).
- Can be made available for a role by assigning data from other information.
- Can be used to determine the decisions and actions to be taken within a choreography.

The *variableDefinitions* construct is used for declaring one or more variables within a choreography block.
For more detailed information on the different features of variables please refer to the WS-CDL specification. Here we point out just one feature which can be relevant to the rest of the presentation and for the Grid4All free attribute.

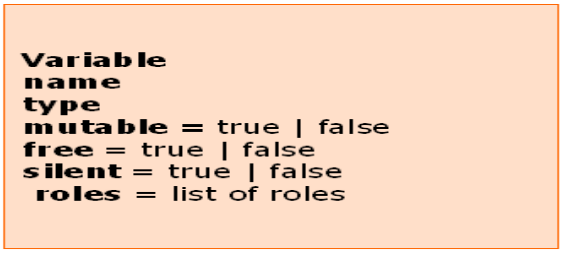

Figure 11: The WS-CDL variable element

The free attribute when specified to true make the variable free. A free variable of a choreography plays a role of choreography parameter. So to use such choreography one must bind the free variable within effective variable. This is useful to create choreography by composing other parametrized one.

C) Token and token locator

Tokens are a set of data used to identify either a role or a specific instance of role; while a role defines Web service many instances of a given roles can co-occurs within one participant implementation. Tokens and tokens locators joined which channel (endpoint address) identified specific subset of exchanged data that grantees the uniqueness of values between instances and so identify specific instance of a role. For example, each market created by invoking the market services role must be identified by the other partner instances (can use a fresh generated identifier for each instance). Token and token locator correspond to correlation data in the BPEL specification.

4.2.4 The collaboration model

The collaboration model corresponds to the definition of the global collaboration protocol. The global protocol is expressed in term of data oriented and structured interactions. The data allow to specify partner state oriented constraint while the structured activities specify a set of order and time constraints on the partners interaction.

A) Interaction, assign and silent action

As claimed before choreography is about a global view of what should happen between partners (participants). The definition of a collaboration between two participants passes throw interaction. While we are in a Web service context, interaction correspond to Web services invocations (a dually provided services operation) between participant roles. An interaction is between two roles. It happens on a specific channel and it corresponds to an operation (the operation model must fit the channel model). As explained before WS-CDL specification allows global observed data, the interaction can (optionally) specify the involved data variables during the exchange (can be variable or channel variable). Figure X represents an interaction definition.

```
<interaction name=
channelVariable="qname"
 operation="ncname"
 time-to-complete="xsd:duration"?
  align="true"|"false"?
  initiateChoreography="true"|"false"? >
 <participate relationship="qname"
          fromRole="qname"
          toRole="qname" />
 <exchange messageContentType="qname"
        action="request"|"respond" >
  <use variable=""/>
  <populate variable=""/>
 </exchange>*
 <record name="ncname"
       role="qname"
       action="request"|"respond" >
 </record>*
</interaction>
```

Figure 12: the WS-CDL interaction element

Here we detail three points which are non intuitive:

- The **align** attribute : when set to "true" it means that interaction results in the common understanding of the messages exchanged at both endpoints are specified in **fromRole** and **toRole**.
- The **initiateChoeography** attribute : when set to **true** this mean that the result of this interaction is a new instance of **toRole** service will be created and detached from the current instance (or thread) by this interaction .
- The **record** element expresses the impact of the exchange on the state variable of the two roles.

WS-CDL offers also two other basic activities: the **assign** activity and the **silent action.** The **assign** is used to data transfer from variables or states to others. While The **silent action** specifies that non observable located activities can be executed within a specific role. The silent action is an abstraction of an internal activities. Internal activities represents either internal data manipulation or communication activities (eg. invoking services) in the last case the interaction must be with partner other than the choreography ones.

B) The structured activities

Basic activities are structured by a set of structure constructors (workflow like), they specify the accepted order of the basic activities (**interaction**, **assign** and **silent action**) to be executed with the choreography. It defines the order constraint of the collaboration. The order of interaction is defined from a global point of view (no central coordinator).

- **Sequence** : all the enclosed activities must be executed in the order of their appearance. e.g. the registration interaction between the bider and the market must be done before the bider sends its bid (or play negotiation) to the market.
- **Choice** : the choices can be guarded by either defined condition or undefined condition. The condition when expressed this mean that the interaction are guarded by populated data between all the involved roles. e.g. if the bid value sent by the bider is less than the market value then a “badbid” interaction occurs (from market to bider).
- **Parallel**: all the enclosed activities can happen in parallel which means that all the interleaving combination of interaction is allowed by the collaboration.

C) Define controlled blocks

In addition to the structured constructors, WS-CDL proposes an event oriented constructor **Work unit**. Work unit is a block of a choreography that is guarded by a set of possible events.

Here is the work unit element definition.

```
<workunit name="ncname"
    guard="expression"?
    repeat="expression"?
    block="true|false" >
    Activities
</workunit>
```

Figure 13: The WS-CDL workunit element

- **The guard**: the guard represents an expression event. The guard in WS-CDL can be either availability of data (from null to instantiated value of a variable), condition on variable or a time constraint.
- **Repeat:** defines an expression to re-evaluate the work unit guard. Note that using the guard and repeat one can express different iteration schemas.
- **Block:** when **work unit** guard is evaluated and the guard and the data used are not available then two cases are possible: either to force the process to wait until the data values become available (true value of block) or to skip the work unit (false value of block).

The firing rule of work unit depends of the moment where the work unit is evaluated, the availability of data of the guard and the block attribute value.

4.2.5 Other features

A) Composition

WS-CDL allows a modular choreography definition. With a choreography we can define sub-choreography (parametrized when free variables are used). Those sub-choreographies can be performed by the principal one.

B) Exception and finalization

A Choreography can recover from exceptional conditions and provide finalization actions by defining:

- One **Exception block**, which may be defined as part of the choreography to recover from exceptional conditions that can occur in that enclosing choreography. The exception handling is realized by a set of work units where the guards by condition related to the raise of exception.
- One **Finalizer block,** which may be defined as part of the choreography to provide the finalization actions for that enclosing choreography. Finalizer blocks are also defined by a set of work units. From a conceptual point of view this block can be used for a proper termination in case of exception.

4.3 From WS-CDL to Abstract BPEL : the e-market framework

A global view description in the participant communication behaviour offers conceptual clarity not found in each participant behaviours descriptions, partly because a global interaction deals with the system intended to be realised. Real execution of the description, however, is always through communication among participants which (as the notion of choreography dictates) may involve no centralised control. Thus the question can be raised here about the relation between the choreography specification and its realization i.e the set of a participant behaviours that realise it. Thus We need to bridge the world of global description to participant descriptions. The ultimate goal of the designer of

choreography is to define the desired choreography and that choreography will realized. The question raised here is :
I*s all the WS-CDL production is realisable by a distributed participant projection? If not what are the rules to respect in order to guarantee an automatic methods to derive each participant behaviours that realize such choreography?*
The answer to the the first question is no i.e not all the WS-CDL or the global protocol definition is realizable by a set of distributed interaction participant. Here we present intuitively two cases of unrealisable choreography and also announce the rules to respect.

- **The connectedness:** when we express a sequence (total order) between a set of interactions (global viewed exchanged message), we must insure that this sequence is realizable in case of absence of total control. Let we consider the following choreography: *A —m1—> B*[1] then *C —m2 —>D*. If we project such choreography on each role we find.
- A send m1
- B receives m1
- C sends m2
- D receives m2

If we consider such four participants definition the result is that their no possibility to restrict (in a distributed case) that the communication between A and B happens before the interaction between C and D as claimed by the choreography (the two interaction are in parallel). The idea is that the sequence structure constraint must be insured in the global definition by a causality in control transfer between involved partner (called connectedness).

Well-threadness : always about the causality. Let we consider this case :
A—m1—>B then *B —m2—>C* then *C—m3—>A* then *A—m4—>B*
and let we suppose that the three first interaction provoke the Creation of new instance of the target role . Apparently the causality over roles is respected to realize the sequence execution. But if we analyse the two last interaction the *A* do not correspond to the same (in order to respect causality) instances of the role A and while instance may co-occur in parallel their is no way to restrict that the last interaction to occur immediately after the first (two action of the same A instance). Thus causality must be respected among the instances of roles.

We call a WS-CDL specification that respect those rules as Well-structured.

The WS-CDL Working Group addresses this problems by proposing normal foundation to WS-CDL throw pi-calculus based central communication language and they have proposed an algorithmic method to derive from the WS-CDL description each participant behaviours (from global collaboration protocol extract the local ones so that their composition) [GCL07].
Such a projection is called *endpoint projection (EPP)*. The EPP algorithm proposed is *sound* and *complete*, in the sense that all and only globally described behaviour is realised as communications among participants if the WS-CDL is Well-structured. The soundness and completeness of the EPP is proved only for well-structured choreographies.

This theoretical foundation of WS-CDL and the EPP is one of the central point in our research approach. The designer after defining a well structured Choreography that specify a negotiation mechanism can automatically generate the participant behaviours (Abstract BPEL description) that represent the minimal set of behavioural and data constraint that must be respected by any actor desiring playing a role. The EPP then ensures that the interaction between a set of actor that implements correctly the resulted roles specification then their interaction will behaves exacted as

[1] *A—m —>B* stands for an interaction between A an b (A sends m to B)

specified in the choreography specification (i;e the mechanism).This guarantees that only valid market according a mechanism will be allowed in the market place.

An additional advantage is that the theoretical foundation of the WS-CDL offer a starting point toward model generation and then model checking of the mechanisms. The fig 13 represent the relation between the functional model presented in the section 2 and the frame work (composed by WS-CDL and abstract BPEL) that realise it. The figure instantiate MSL by the WS-CDL language and the RSL by abstract BPEL.

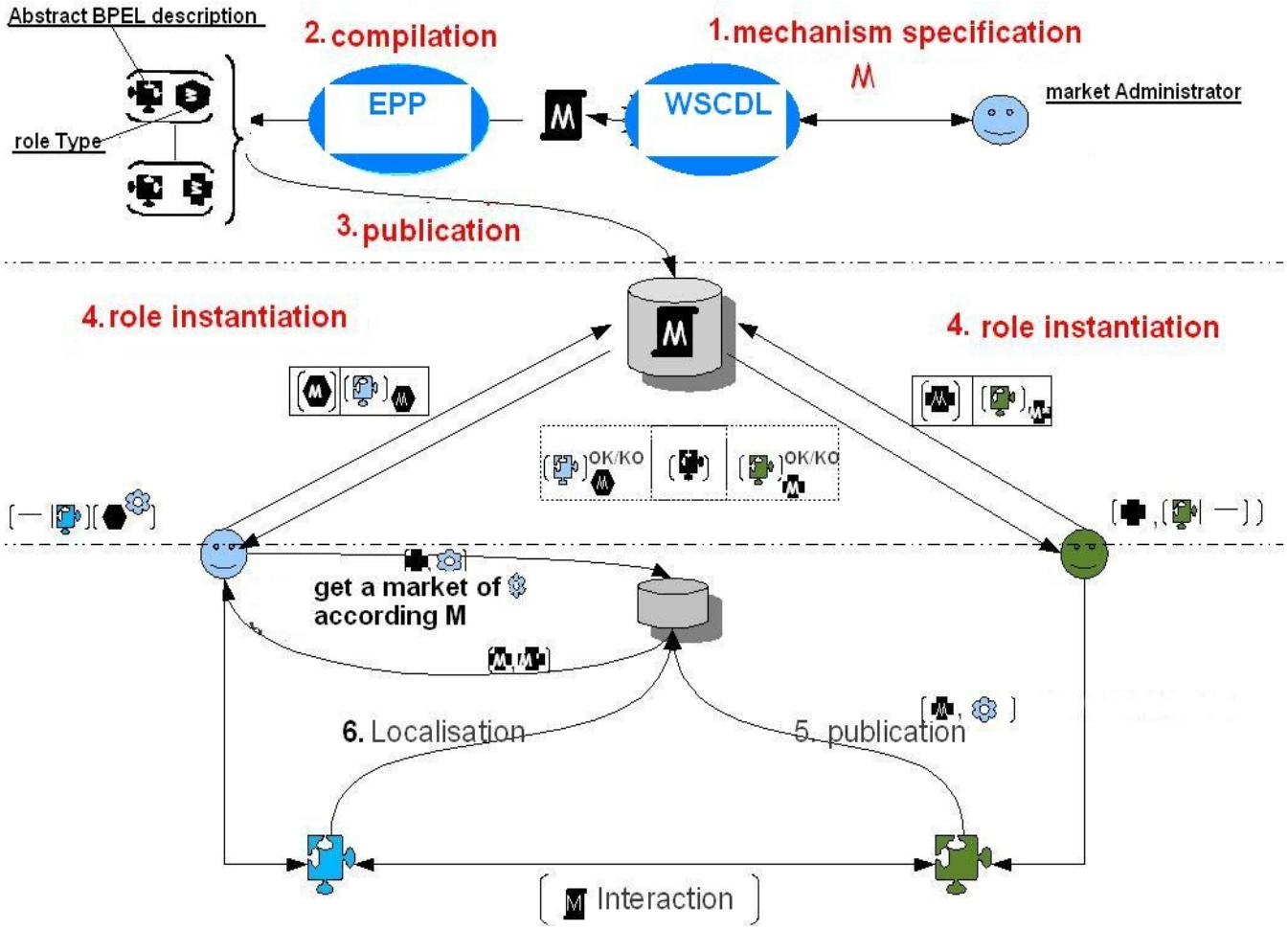

Figure 14: The e-market functional model and Web services Framework

5 About the relevance of the proposition to the grid4all project.

The CRE is about proposing an architectural solution to a negotiation e-market and also a methodology to manage the e-market in open environment and without allowing *ad hoc* market to appear. Our proposition is based on three points:

SOA as an architecture of the e-market: the originality is to consider the negotiation process as a business process involving a set of peer to peer partners. We have proposed a functional model compatible with the SOA one and working in two levels. The mechanism level where the service is a mechanism that allows the e-market to be enriched by the set of accepted negation protocols. The instance level or market level where the market is a distributed composition of a set of actors that validate a mechanism. We have also shown how SOA is suitable for open environment.

Web service framework to implement the functional model: web service technologies offer a framework for loose-coupling software components. This framework offers set of interface description languages and also composition languages. Two languages retain our interest:

- **A peer to peer composition language WS-CDL** : this specification language offers a set of constructors in order to specify a multi-partite interaction between a set of Web services. It defines such collaboration protocol from a global point of view (without adopting particular participant point of view). The WS-CDL language presents a specific interest for the GRID4All project for three main reasons:
  - A global description of communication behavior arguably offers conceptual clarity. The natural perspective of a global behavior ensures that the common collaborative observable behavior is not biased towards the view of any one of the participants. This make its implementation independent.
  - Typed nature of the allow to identify or to specify specific interaction between partner.
  - The language can support also global behaviors based on exchanged data values. This is a central point to complete the negotiation mechanism description by specifying constraints (or data oriented rules) on interaction (e.g. a session bid value must be greater than the session open value otherwise a reject message is sent back to the participant)
- **An interface description language Abstract BPEL**:. Such language is used to describe the access rule (order and time constraints) to use a long-running Web service (BPEL services). The behaviours of a service is an abstraction that allow a separation of web service interaction protocol and its implementation. The Abstract BPEL is relevant for the grid4All project to describe in machine understandable way the different constraint that an actor must respect to realise a role in given mechanism (the actor must be an implementation of the role).

The first language allow the design of a mechanism in a simple an intuitive manner and the second describes the role types definition. We have also pointed out the existent of a theoretical foundation behind those languages that guarantees a full automatic translation from a global specification (a mechanism) to the minimal set of constraints that define each role. This transition is possible for well-structured choreography. To reach a Well structured choreography the designer must respect a set rules.

Control of the validity of the market

Market actors can then check automatically their compatibilities (abstract of their behaviours) and the role specification in an automatic way.

The next deliverable is dedicated to this last point. We will illustrate our methods on a negotiation protocol and show the use of tools to do so.

Bibliography

[TYC05] M. Feldmanand K. Lai, A Price-anticipating resource allocation mechanism for distributed shared c, 2005

[SHR03] Fu Y., Chase J., Chun., Schwab S., and Vahdat A., SHARP: An Architecture for Secure Resource Peering, 2003

[SOA03]: M. P. Papazoglou., Service-Oriented Computing: Concepts, Characteristics and Directions, 2003

[GLO02] Talia, D., The Open Grid Services Architecture: where the grid meets the Web, 2002

[SDL07] R. Chinnici, H. Haas, A. Lewis, J-J. Moreau, D. Orchard, S. Weerawarana, Web services Description Langue V 2.0, 2007

[BPE07] Web Services Business Process Execution Language Version 2.0, 2007

[HAD05]: S. Haddad, T. Melliti, P. Moreaux and S. Rampacek, A dense time semantics for Web services specifications languages, 2005

[CDL05] W3C Working Groupe, Web Services Choreography Description Language V1.1, 2007

[GCL07] R. Milner, G. Brown, S. Ross-Talbot, M. Carbone, K. Honda and N. Yoshida, Theoretical Basis for Communication-Centred Programming, 2007.