Adaptive Support of Inter-Domain Collaborative Protocols using Web Services and Software Agents

Adomas SVIRSKAS [a,1,2], Michael WILSON [b], Bob ROBERTS [c] and Ioannis IGNATIADIS [c]

[a] *Enterprise Communications Department, Institut Eurécom, France*
[b] *CCLRC Rutherford Appleton Laboratory, UK*
[c] *Kingston University London, UK*

Abstract. This paper explores the challenges of constructing an architecture for inter-organisational collaborative interactions based on Service Oriented Architecture (SOA), Web services choreographies and software agents. We present an approach to harmonisation of the "global" or neutral definition of business collaborations, with partner-specific implementations, which can differ in terms of platform, environment, implementation technology, etc. By introducing the concept of pluggable business service handlers into our architecture we draw on the work carried out by ebXML initiative, business services interfaces, in particular. Due to increasing need for better management of collaborative interactions, Virtual Organisations (VO) become an important tool for creation and maintenance of federated trust domains among the collaboration partners. We look into the software agents abilities to serve as the background support mechanism for the automation and management of the Virtual Organisations lifecycle.

Keywords. B2B Collaborations, Web services, Choreography, WS-CDL, Agents

Introduction

There is a growing need for flexible and adaptive business protocol support in Service Oriented Architecture (SOA) based e-business solutions and those based on Web services technology in particular. Recent developments in Web services field provide promising opportunities for integrating data, applications and business processes. The latter, however, is the most complex case of integration as it requires strong support for both business process semantics and technical infrastructure in order to tackle heterogeneity at all levels. According to Wombacher et al. [1], in today's B2B solutions landscape loosely coupled business processes are quite rare. Despite many promising advancements, Web services technology faces a number of issue to address heterogeneity at different levels of integration. Usage of simple stateless Web services is not sufficient for implementing business processes while static binding does not use full potential of loosely coupled systems and the SOA advantages [1]. In order for

[1] Corresponding Author: Adomas Svirskas, Enterprise Communications Department, Institut Eurécom, 2229 route des Crêtes - BP 193, 06904, Sophia Antipolis Cedex, France

[2] The work presented in this paper was done while working at CCLRC Rutherford Appleton Laboratory and Kingston University London, UK

distributed services to achieve common meaningful business goals, some rules of engagement are necessary. These rules govern the interactions between the services (referred to as services conversation) by defining message exchange sequences, message formats, roles of the collaboration participants etc. A set of service conversation rules is also referred to as business collaboration protocol or simply business protocol.

Business protocols can be made modelled using business process modelling tools and made available in machine-processable format through choreographies – declarative descriptions of service conversations. Vinoski [2] argues that Web services choreographies must take business processes into account: trivial Web services solve only trivial issues; non-trivial Web services must play a part in business processes. Business-to-business integration (B2Bi) requires standardized choreographies, i.e. definitions of the "conversations" between cooperating applications that allow them to work together correctly [2]. Simply put, choreography is a model of the sequence of operations, states, and conditions that control the interactions involved in the participating services [3]. The interaction prescribed by a choreography results in the completion of some useful function. Examples include the placement of an order, information about its delivery and eventual payment, or putting the system into a well-defined error state. Gortmaker et al. have presented an extensive of choreography definitions in [4] along with a thorough discussion of choreography and orchestration.

The rest of this paper is structured as follows. Section 1 discusses business protocol support issues and adaptivity dimensions, Section 2 explains the roles of choreography and orchestration in collaborative multi-party interactions, Section 3 provides a motivation for a business service handler-based approach to connect the public choreography to services and, in turn, private operations. Section 4 describes the main concepts of our approach and relates it to ebXML framework and Section 5 discusses applicability of software agent technology in dynamic Virtual Organisations (VO), followed by the conclusions section.

1. Business Protocol Support Issues

There are a number of requirements to be taken into account and satisfied in order to make the idea of commonly agreed business protocol specification and their subsequent enactment viable, let alone efficient, robust, scalable and secure. These requirements arise both from the business and technology domains and represent quite a wide spectrum of issues ranging from the field of activity of a business analyst to that of a software developer involved in business application coding.

Our paper aims to contribute to the aspect of adaptivity of business protocol support by collaborating partners. This aspect is a part of a broader issue of adaptivity in the business collaboration domain. Adaptation, which is defined as "*modification of an organism or its parts that makes it more fit for existence under the conditions of its environment*" [5], and the speed of it are crucial factors which determine the success and longevity of any business subject or formation in a constantly changing business environment. In the context of B2B collaborations adaptivity has several flavours: adaptivity of the business models to different business requirements and environments (Hofreiter & Huemer discuss this in [6]), adaptivity of business protocols in response to business models' changes, adaptivity of the partners' end-point services to the changes in the business protocols descriptions.

In addition, the business protocols should be adaptive to the changes of the partners enacting the roles defined in the protocols (both choreography and orchestration should support this). These requirements are contradictory in many cases and cannot be efficiently addressed in isolation: for example, optimisation of the work of a business analyst does not necessarily results in scalable implementations of end-point services and/or their faster time to market. Thus, a holistic approach is needed to business and collaboration modelling, enactment and process management in order to increase overall B2B collaboration adaptivity. **Figure 1** depicts the circular dependencies between the main adaptivity dimensions and emphasizes the need for balance between the different views.

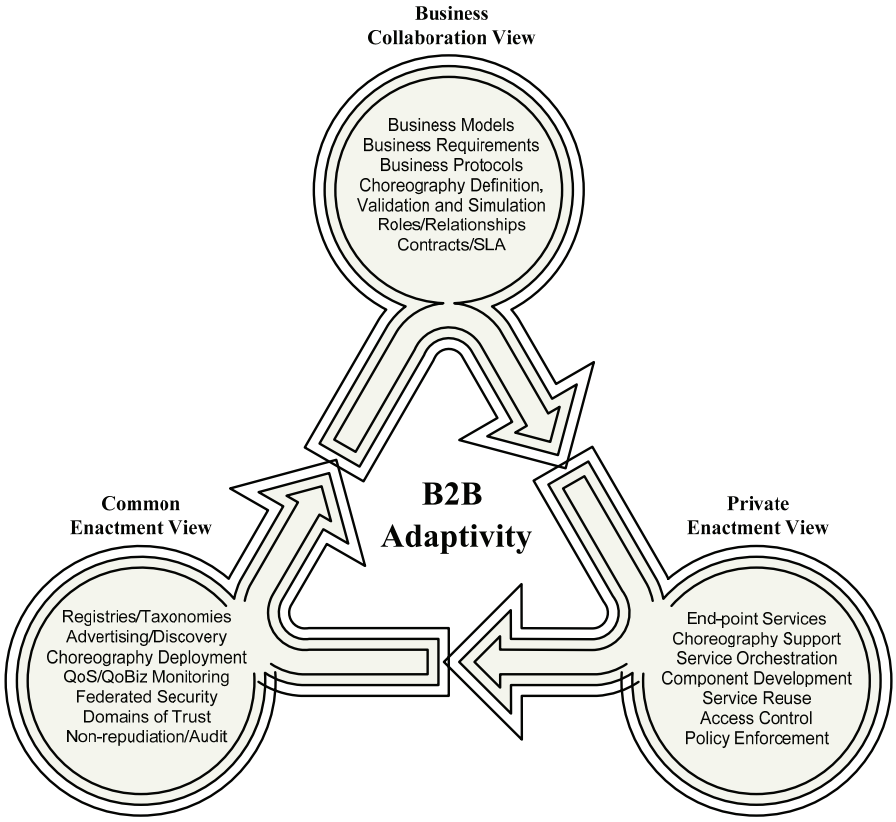

Figure 1. Relationships between Different Adaptivity Views

This kind of approach puts quite high requirements on the supporting IT infrastructure: not only the business modelling, choreography support, service development and deployment tools and established practices are needed, but also a coherent unifying framework, preferably based on open standards, should be established. Once the choreography definition is created, it needs to be deployed somewhere (the SOA suggests use of a registry) and advertised for reuse. It needs to be

distributed to (or perhaps, depending on the overall interaction model, discovered by) the appropriate collaboration participants (dynamically chosen to play certain applicable roles) and accepted by each of them as a contract for subsequent interactions. The choreography script is then enacted by the participants' end-point services and this process should be monitored and managed by user and administrative tools. In addition to the basic requirements mentioned above, there are requirements for security and trust [7][8], business-level contract [9] support, adequate levels of Quality of Service (QoS) and Quality of Business (QoBiz) by specifying, negotiating, monitoring and enforcement of Service Level Agreements (SLA) [10]. These and some other requirements are defining characteristics for B2B solutions and Virtual Organisations to be accepted by the corporate world [11].

It is therefore clear, that adaptivity in the business protocol support area is a complex issue and depends on many, frequently conflicting, factors. Cherinka et al. characterises complex adaptive systems in [12] as "*dynamically assembled systems characterized by multiple competing stakeholders, fluid requirements, emergent behaviour, and susceptibility to external pressures that can cause change across the entire system*". This statement was used to reflect the nature of net-centric operations of the US Department of Defence; however it is applicable to the area of dynamic B2B Virtual Organisations, which also must accommodate unpredictable external factors that demand rapid response and flexibility to change [12].

2. Service Choreography vs. Orchestration

For the sake of clarity we would like to note the difference between choreography and orchestration, as these two terms are being used improperly sometimes, which causes confusion. Orchestration specifies the behaviour of a participant in a choreography by defining a set of "active" rules that are executed to infer what to do next, once the rule is computed, the orchestration runtime executes the corresponding activity(ies). Orchestration assumes existence of an entity, which is the central point of control and governs overall workflow of activities, effectively composing a new service from existing services. The standardization of orchestration and the emergence of a new application model will also benefit from a robust B2B layer, such as ebXML, in the Web services stack.

As a matter of fact, orchestration could take its full dimension from the extension of the business semantics to the application model [13]. Choreography, as explained before, is meant to be enacted by peers without an intermediary, at runtime, the choreography definition can be used to verify that everything is proceeding according to plan. Choreography can also be used to generate a public interface (e.g. abstract BPEL) that can be used to tie in internal activities to support the choreography [13]. Dubray also differentiates in [13] between the two concepts by arguing that choreography defines the fabric of an SOA while orchestration, helps to build "processing entities" - non-trivial services, which can perform tasks, needed to support complex business interactions.

Ross-Talbot [35] proposes a concept of behavioural backbone on top of a choreographed distributed infrastructure where the interactions among parties are described using a declarative language. It is possible to generate monitors for each of the roles, which allow gathering run-time information without affecting the behaviour of the services. The behavioural backbone provides the monitoring agents and the

consolidation of that monitoring information against choreography. It allows observing non-intrusively potential collaboration deviations from the initial collaboration plan, which allows understanding the collaborative processes and managing them. Of course, such agents can serve many different purposes – security, transaction support, logging/auditing etc. Our approach based on business service handlers and application level gateways is similar and takes into account the whole range of issues beyond monitoring. The software agent technology discussed in the Section 5 is in particular suitable for supporting such meta-protocols for monitoring and managing the collaborations.

To summarise this discussion, **Figure 2** depicts different domains of control and inter-relation between the choreography and orchestration in business collaboration context.

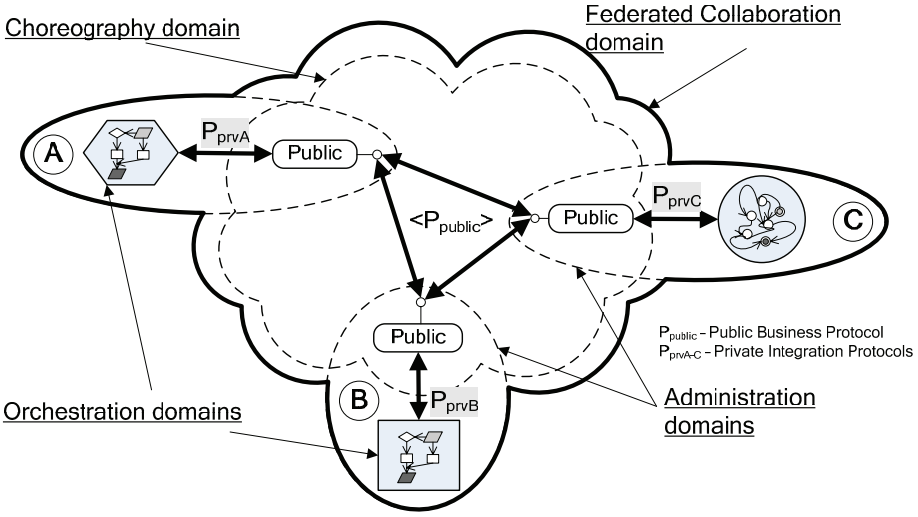

Figure 2 - Web Service Choreography vs. Orchestration

We can distinguish four conceptually different domains here:

- *Administration domain* contains the resources belonging to an organisation or a large department, these resources are centrally managed according to corporate policies
- Within each administration domain typically exist several *orchestration domains*, which correspond to the established internal workflows and/or are used for service composition
- The goal of collaborating parties is to establish a *federated collaboration domain* (Virtual Organisation) using commonly understood interaction protocol policy-driven access control, trust management etc.
- One of the main instruments establish and maintain such federated collaboration domain (we can call this process VO management) is choreography support – from modelling of the interactions to deployment and enactment of the business protocols expressed by means of

choreography descriptions. This is achieved by forming a choreography domain consisting of public services exposed by the collaboration partners. These services act as gateways to virtualised business services of the partners required to achieve the outlined business goals. In addition to passing the incoming messages during choreography enactment to the appropriate internal service endpoints, gateways are able to manage and monitor the interactions among the services by reasoning upon and enforcing the necessary policies. This style of interaction adheres to the managed public process e-business integration pattern [36].

The considerations mentioned above and some earlier developments, such as Web Service Choreography Interface (WSCI) [14], ebXML Business Process Specification Schema [15] served as input for W3C to establish the Web Services Choreography Working Group and begin work towards a language that can be used to describe collaboration protocols of cooperating participants, which act as peers and their interactions may be long-lived and stateful. Web Services Choreography Requirements document [16], provides the following definition of choreography: "Web Services Choreography concerns the observable interactions of services with their users. Any user of a Web Service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. A specific set of interactions maybe related over time to some form of collaboration grouping that is initiated at some source and runs through a set of Web Services and their client. A choreography description is a multi-party contract that describes from global view point the external observable behaviour across multiple clients (which are generally Web Services but not exclusively so) in which external observable behaviour is defined as the presence or absence of messages that are exchanged between a Web Service and it's clients" [16]. A notable contribution to these requirements was [17], which advocated the need for complementary but separate languages - choreography programming languages and choreography description languages. Goland showed in [17] rather clearly, using motivating use cases, the difference (from choreography point of view) between executable languages such as Java, C#, BPEL and similar, and declarative description languages, which capture a global view of messaging activity and are not designed to provide information about how participants implement their individual tasks. Goland also explained in [17] the need for generating role-specific code skeletons from choreography description in order to facilitate faster and more convenient implementation of individual functionality. The choreography description language uses roles to differentiate between the participants in choreographies. We will discuss this aspect in greater level of detail in subsequent sections.

3. Adaptivity Issues in Choreography Life-Cycle

The result of the mentioned W3C WS Choreography Working Group effort is WS-CDL language [18], which is the means to define a technical multi-party contract, mentioned above. WS-CDL specification is aimed at being able to precisely describe collaborations between any types of participants regardless of the supporting platform or programming model used by the implementation of the hosting environment, thus addressing heterogeneity issues [18]. Choreographies must also completely hide component-level implementation details. Moreover, the same choreography definition

(potentially involving any number of parties or processes) needs to be usable by different parties operating in different contexts (industry, locale, etc.) with different software (e.g. application software) [18].

Choreography definition using WS-CDL allows building of more robust services because they can be validated statically and at runtime against a choreography description, verification absence of deadlocks and live-locks, etc. It also helps to ensure effective interoperability of services, which is guaranteed because services will have to conform to a common behavioural multi-party contract, mentioned earlier [19].

However, compliance of the participating services to the common contract might result that the choreography enactment is hard coded into the implementations of the services and/or their composition mechanisms. This approach poses a two-fold problem: it reduces reusability of the services and also makes it difficult to change choreography description without a need for massive programmatic changes at the participating end-points. Goland [17] discusses these issues from developers' point of view in his contribution to WS Choreography requirements.

A possible alternative to global-contract choreography is a technique called mediation, where an intermediary agent is involved in communication between parties and ensures compliance of message flow to expected/requested behaviour of each party. A notable example of such approach is Web Service Modelling Ontology (WSMO) Choreography [20]. WSMO is an ontology for describing various aspects related to Semantic Web services [21].

The WSMO framework provides support for choreography and orchestration as part of interface definition of a WSMO service description. An interface describes how the functionality of the service can be achieved (i.e. how the capability of a service can be fulfilled) by providing a twofold view on the operational competence of the service: Choreography decomposes a capability in terms of interaction with the service (service user's view) Orchestration decomposes a capability in terms of functionality required from other services (other service providers' view) With this distinction different decompositions of process/capabilities are provided to the top (service requester) and to the bottom (other service providers). This distinction reflects the difference between communication and cooperation. The choreography defines how to communicate with the web service in order to consume its functionality. The orchestration defines how the overall functionality is achieved by the cooperation of more elementary service providers [22]. WSMO choreographies are based on the Abstract State Machines methodology. Cimpian & Mocan in their work [23] describe a process mediation approach based on WSMO choreographies and Web Service Execution Environment (WSMX) [23], a reference implementation for WSMO.

4. Suggested Approach

In this paper we propose software components called handlers to represent the logic of harmonizing public choreographed processes with private functionality of end-point services. Handlers are registered with and coordinated by a choreography support service, which, in turn, used by the end-point services to support global choreography contracts. Configured with pluggable handlers choreography support service mediates two-way message exchange between the "outside world" and the local processing entities. Based on the available handlers, various request types and formats can be routed, translated, and fulfilled by the business services. Choreography support service

can relatively easily be reconfigured, adapted, and extended as new processing entities need to be supported. In addition, dynamic selection of processing entities to play the prescribed roles, policy enforcement, trust and security support and other non-functional tasks can be performed by the handlers. The handlers can be implemented as a chain of message filter put in front of processing entities deployments. A handler takes a choreography definition, the role, which processing entity is supposed to play and maps the choreography messages to local operations at run-time.

4.1. The Role and Status of the ebXML

We call the handlers business service handlers, drawing a parallel with the naming used in ebXML framework [24]. The original name for these components in ebXML framework was Business Service Interface (BSI), which can be described as a piece of software that handles incoming and outgoing messages at either end of the transport [25]. The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable e-business. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics. Dubray explains the purpose of ebXML BSI in his overview of ebXML [24]: *"The Business Service Interface (BSI) should enforce the business collaboration protocol (ebXML BPSS). At any point in time, the BSI is able to determine if a message makes sense from a business perspective (is format correct? did it come on time? in the right sequence? ...). The BSI may be directly communicating with an application, but it is certainly wise to use a broker that will dispatch ebXML requests and responses to and from your business applications. Typically, this broker is going to be a business process management system."* This explanation actually outlines the core functionality of BSI and justifies the need of pluggable brokers to support a variety of business process management systems. The OASIS ebXML Business Process (ebBP) Technical Committee (TC) later discussed a possibility to differentiate between Business Services Interface and Business Service Handler (BSH) in order to separate the abstract interface from its implementations. By proposing the name change to business service handler, the ebBP committee harmonised the naming between the business and messaging domains – the ebXML Message Service Specification [26] defines the Message Service Interface and Handler separately.

In ebBP Specification v 2.0 [27] the BSI is defined from a different perspective: as a logical definition for a party's actions, exposed as business services. It may be seen as a logical shared definition at different nodes. Logically, a BSI is a partner's implementation of the shared definition of business states and actions relevant to a common business goal. The BSI specifies the allowed set of business process and business object states of a business process, and the rules governing transitions between those states. In the context of the ebBP technical specification, only the shared business process is being managed. The interface to the BSI is through business messages and signals [27]. This defines the functionality of the BSI closely to the functionality of an individual partner required to support WS-CDL based common multi-party contract. Therefore the ebXML BPSS and W3C WS-CDL define substantially similar approach to enactment of common business goal and idea of pluggable business handlers follows this paradigm.

The ebBP technical specification does not, however, specify how the BSI is implemented. For example, the BSI may be enabled through a BSI-aware business application or through behaviour implemented as a part of a Message Service Interface

component. The business application may business signals that are sent (realized) by the Message Service Handler [27]. Similarly, WS-CDL [18] does not specify how collaborating parties implement/map their services to comply with the common contract. We think that it useful to turn to the ebBP TC work when architecting choreographed Web Services solutions, as the ebBP v2.0 specification takes Web Services into account and explicitly relies on choreographed collaborations (no relation to WS-CDL is defined yet; the ebBP TC, however, is working on ebBP and WS-CDL layering). An ebBP Choreography is an ordering of Business Activities within a Business Collaboration and is specified in terms of Business States and transitions between those Business States. Execution of the backend systems, which instruct the BSI to send or receive messages, advances the state of a collaboration. Similarly to WS-CDL, there is no execution engine associated to the collaboration itself. Although WS-CDL and ebBP address similar problem domains, the divergent foci of the two enables them to be layerable - while WS-CDL focuses primarily on the web service perspective, ebBP describes the pure business message flow and state alignment. As such they are not mutually exclusive.

Barreto [28] argues that WS-CDL and ebBP could be used in a loosely coupled, yet complementary manner, where WS-CDL supports the choreography based on endpoint references related to WSDL, while ebBP specifies the operation mapping to the recognized business transaction patterns. This association is beneficial and useful where complex activities occur in the collaboration environment.

4.2. Operation Mapping

One more notable aspect of ebBP v2.0 specification is mapping of Business Transaction patterns to abstract operations through the OperationMapping constructs (still work in progress) [27]. An operation mapping specifies a possible mapping of a business transaction activity to a set of Web Services operation invocations to enable the participation of non-ebXML capable business partner in an ebXML relationship. An ebBP definition does not itself contain a reference to a WSDL file, but rather references to operation names which can be deferenced with specific WSDL files specified at the Collaboration Protocol Profile [25].

The goal of the operation mapping is to offer a flexible mapping scheme to map all document and signal interchanges to any combination of Web Services operation interactions. The mapping is also designed to define an operation mapping on both sides of a Business Transaction Activity (BTA). BTA represents the performance of a Business Transaction within a collaboration and is similar to WS-CDL interaction. This means that the ebBP specification can be used to define the abstract behaviour of complex collaborations between Web Services even in the case where no role in the collaboration is capable of ebXML [27].

The concepts found in ebBP v2.0 specification, therefore, provide several benefits for mapping choreography contracts to the end-point implementations:

- There are clear signs and concrete steps to support Web Services at WSDL operations level. The use of run-time correlation and endpoint references based on emerging addressing mechanisms such as WS-Addressing, WS-MessageDelivery etc. is recommended.
- The ebBP operation mapping is designed to support not only Web Services but other implementation techniques as well.

- The ebBP v2.0 specification suggests flexible approach to operation mapping (and common contract enforcement, in turn) by allowing operations to be mapped both in collaboration description and the partner's endpoints.

Therefore, ebBP [27] serves as a blueprint for architecting choreography-based solutions in general, not only ebXML compliant systems. The ebBP specification is being created (or closely watched) partially by the same individuals as WS-CDL specification, which creates synergy between the two efforts. We took into account many ideas described in ebBP specification and this will greatly help in our future work on this topic.

As we can see from the discussion above, changes in business protocol and choreography description may be quite costly for the end-points to support, therefore some intelligence is needed to allow smooth and fast propagation of the protocol changes. In this paper we propose to introduce an application-level gateway between the public business protocol (choreography) and the end-point composite services. This gateway is a part of the implementations of the collaboration participants' end-points. The functions of the gateway range from virtualisation of the business services of the end-point to choreography support via role-based decomposition of it and mapping of the incoming messages to the appropriate processing entities within the end-point.

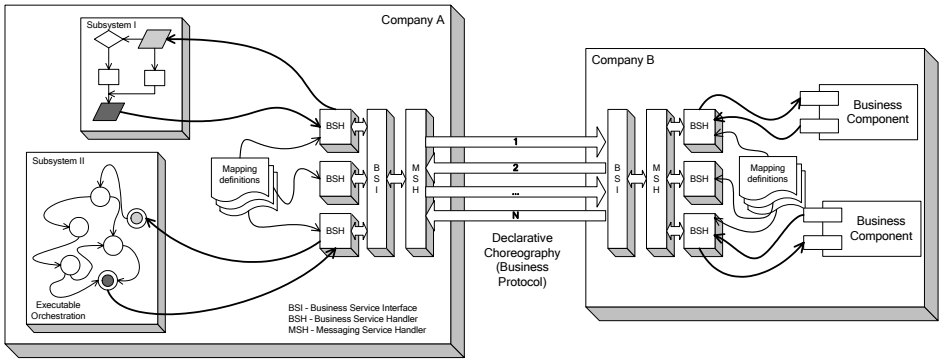

Figure 3 – Architecture of Collaboration Protocols Support using Gateways and Handlers

As shown in **Figure 3**, Inside the gateway there is a set of software components called handlers, which perform distinct tasks, for example to represent the logic of harmonising public choreographed processes with private functionality of end-point services. There are two kinds of handlers – application-specific and application-independent [29]. The latter are reusable across the range of different applications, while the former depend on a particular system. Furthermore, the application-specific handlers are divided into supplier handlers and consumer handlers. Supplier handlers are triggered by the Reactor upon receiving an incoming message and their task is to find an appropriate consumer handler for the message to be processed. Locating of an appropriate consumer handler may be message content and choreography status based.

In order to organise the structure and the operation of the gateway component in a scalable and efficient manner, it is advisable to follow well established patterns for such applications. Schmidt has proposed usage of several patterns (Reactor,

Component Configurator, Acceptor-Connector and other) in application-level gateways in [29]. Other patterns, such as chain of responsibility, also apply here.

The next section describes how choreographies and their adaptive support can be used in VO Management context.

5. Using Software Agents in Virtual Organisations Support

5.1. General observations

Virtual Organisations (VO) are closely related to business collaborations between the services, organisations and individuals involved and are intended to facilitate, directly or indirectly, business solutions in most cases. VO management process can be perceived just as a type of business collaboration or (process) that uses the same mechanisms as for "operational" business collaborations (or processes). The collaboration agreement of a VO specifies processes related to the administration of the VO itself, such as changes to the VO membership or the collaboration agreement [11].

The software agent technologies are suitable mainly for domains of highly complex problems and systems with widely distributed information sources, domains with dynamically changing environment and problem specification, and for the integration of a high number of heterogeneous software systems [30]. Therefore, the agent technologies are suitable for usage in a Virtual Organization, where the participants are geographically distributed, usually with heterogeneous software systems, and where the environment is dynamically changing in response to market needs and requirements.

(Software) agents are autonomous, which is very desirable in unknown scenarios (which usually tend to appear in the real world), where it is difficult to control directly the behaviour of complex business collaborations. Even though it is possible to encapsulate some behaviour by specifying “private” methods, agents must decide by themselves whether to execute their methods according to their goals (agents must be proactive), preferences and beliefs. Agents are also flexible, they have to learn from, and adapt to, their environment. This is important, since when designing an agent system, it is impossible to foresee all the potential situations that a particular agent might encounter, and specify agent behaviour optimally in advance. This kind of situation is highly probable in the most of non-trivial VO interactions.

In a VO setting, a multi-agent system can be employed for supporting internal processes (intra-enterprise level) of the enterprise (e.g. planning and control, resource allocation, production process simulation, and on the other hand, it can support cooperation and negotiation among enterprises (extra-enterprise level) across a value chain (e.g. customers, suppliers, material and service providers, etc). Both types of agents can coexist in an organization.

In addition to the external-internal dimension of agents’ classification, there is another one, which is based on the *specific purpose* of the agent services. The reason for making this distinction explicit is the fact that the business services themselves can be considered as agents as they satisfy most of the agents’ characteristics. Maximilien & Singh [31] in their work of cataloguing Web services interaction styles argue that viewing services as agents enables us to augment the interaction styles of Web services as interactions between and among service provider agents and service consumer

agents. Therefore, it is important to denote the areas of responsibility of the business services and the supporting agents.

As we have explained above, Web services are characterized not only by an interface but also by the business protocols (choreographies) they follow. While business protocols are application specific, much of the software required to support such protocols can be implemented as generic infrastructure components Alonso et al. For example, the infrastructure can a) maintain the state of the conversation between a client and a service, b) associate messages to the appropriate conversation, or c) verify that a message exchange occurs in accordance to the rules defined by the protocols (for example WS-CDL). Part of the task of the infrastructure is also the execution of meta-protocols, which are protocols whose purpose is to facilitate and coordinate the execution of business protocols. It is convenient to think of the agents as the meta-protocol enablers, paving the way for the business services.

For example, before the actual interaction can begin, clients and services need to agree on what protocol should be executed, who is coordinating the protocol execution, and how protocol execution identifiers are embedded into messages to denote that a certain message exchange is occurring in the context of a protocol. In the Web Services domain, WS-Coordination is a specification that tries to standardize these meta-protocols and the way WSDL and SOAP should be used for conveying information relevant to the execution of a protocol [32]. In the Multi-Agent System (MAS) domain, there are other protocols for agents' interaction, which can be useful for implementing the meta-protocols.

Having distinguished between the agents and the services, we need to make sure that these two types of entities coexist peacefully within a single architecture and interoperate properly. Maximilien & Singh [33] propose a framework that augments a typical Service-Oriented Architecture (SOA) with agents. Their principal idea is to install software agents between service consumers and each service that they consume. These consumer service agents expose the same interface as the service. However, they augment the service interface with agent-specific methods. The consumer communicates its needs via the augmented agent interface. Service method invocations are done via the service agent who, in turn, monitors and forwards all calls to the selected service. Both business and meta-protocols can be modelled, validated and verified using the WS-CDL language and tools.

A good example of a consistent set of meta-protocols is the VO Management domain, where the business collaboration partners (peers) interact by the rules agreed by all the VO members and VO managers. These rules are enacted partly by direct interaction between the peers, and partly by the peers and the VO Management. We discuss Virtual Organization Management in the next section.

The concept of multiple agents can also be useful in general-purpose Web service composition. Maamar et al. [34] present an agent-based and context-oriented approach that supports the composition of Web services. To reduce the complexity featuring the composition of Web services two concepts are put forward in their work, namely, software agent and context. During the composition process, software agents engage in conversations with their peers to agree on the Web services that participate in this process. Conversations between agents take into account the execution context of the Web services. The security of the computing resources on which the Web services are executed constitutes another core component of the agent-based and context-oriented approach presented by Maamar et al [34].

5.2. Virtual Organisation Management

A VO typically follows the life cycle consisting of the four phases: a) Identification (Opportunity Identification and Selection), b) Formation (Partner Identification and Selection, and Partnership Formation), c) Operation (Design, Marketing, Financial Management, Manufacturing, Distribution), and d) Termination (Operation Termination and Asset Dispersal). The management of a VO through those phases can be described just as a type of business process (or collaboration) that uses the same mechanisms as for "operational" business processes (or collaborations). The collaboration agreement of a VO then specifies the processes that are related to the administration of the VO itself, for example changes to the VO membership [11].

With regard to the software agent technology, it can bring various advantages to the domain of management of e-collaborations. The technological and integration aspect is covered by the Foundation for Intelligent Physical Agents (FIPA) [38], which tries to maximize interoperability across agent-based applications, services and equipment.

Figure 4 illustrates our proposal for the management of Virtual Organizations using Web Service Choreography and Software Agents. The innovation lies in the fact that whereas Web Service Choreography can be used to coordinate the interactions between Web Services and their consumers, software agents can be inserted in front of those services, and their actions choreographed.

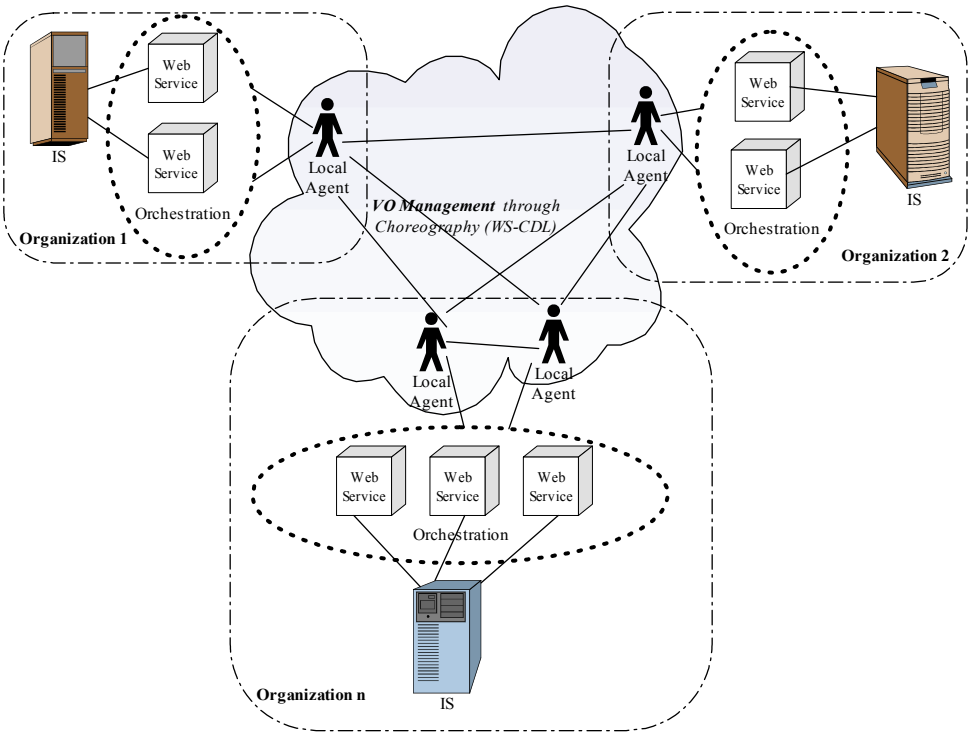

Figure 4 - Virtual Organization Management with Choreography and Agents

Within a Virtual Organization, intelligent software multi-agents can take some of the load in each of the phases of identification, formation, operation and termination of a VO, by automating the relevant processes. Various agent technologies can also be used for the agents' private knowledge, maintenance, specification of various ontologies, and ensuring service interoperability across the value chain.

The proposed solution is a generic one, in the sense that it does not distinguish between the number of agents or their type (e.g. per service, business process, or enterprise). It assumes however, that at least one local agent exists per each organization that participates in a VO. The interaction between the organizations in the VO is carried out with interactions between the respective agents. The latter communicate with the Information System (IS) of the organization via the appropriate Web Services. Whereas within a single organization those Web Services follow orchestration rules, as described earlier, the whole VO is coordinated by choreography rules that are enacted by each of the local agents assigned to an organization. In that process, agents communicate and exchange information with local agents of other organizations. The use of agents adds flexibility to the operations of the VO, whereas at the same time the use of choreography rules ensures the efficient management of a VO without the need for a centralised service.

The process of the creation of a Virtual Organization has its counterpart in the cooperative team creation or coalition formation processes in the agent technologies domain. In this domain, a group of cooperating agents (coalition) is formed to fulfil a common goal. The individual agents are self-oriented and they don't share all information or their intentions. The agent technologies in this case classify the knowledge as public, private and semi-private. This has a high potential for the management of Virtual Organizations, where there is not a central point of control, but the e-collaborations are rather peer-to-peer, in which case it is important for each peer to have control over the availability of its own information to the other peers in the network and restricting access to the confidential data.

Conclusions

In this paper we have proposed an approach to the issue of choreography support in heterogeneous collaborative business interactions. We base our concepts on the assumption that service choreographies can be mapped to end-point operations using either rich service universal descriptions or end-point specific operation mappings. In both cases it is possible to derive programmatically the configuration artefacts, which can be used to configure business service handlers dynamically at the runtime. This possibility is attractive from many points of view, the most important perhaps being clean separation of business services interfaces and business services implementation.

Business service handlers can also be used for various other purposes – policy enforcement, trust and security support, collaboration correctness monitoring, QoS monitoring, transaction logging, etc. Choreography languages, such as WS-CDL can perhaps be enhanced to support declarative specification of the mentioned aspects for subsequent programmatic propagation of these specifications to the service end-points and mapping to the end-point specific mechanisms.

These ideas make basis for future research in this area alongside with detailed design of the pluggable business handler framework, which is described in this paper at conceptual level and many decisions still need to be made. Standard operation

mappings between choreographies and implementation languages such as Java, C#, BPEL are one of the main issues in this area along with rich service description and matchmaking problems. It is a promising sign that the ebXML BPPS specification takes into account these issues and drives the effort to solve them in standard interoperable manner, as support of the open standards is crucial for adoption of solutions.

In the context of our work we use Web Services as the underlying technology to implement of models. The requirements for VO management described here can be met by a subset of the current WS* specifications, but they require secure, stable and interoperating implementations from a variety of IT vendors, which is not the case yet.

We also believe that combination of SOA and Web services in particular with the software agents can be very promising. The agents differ from services in a number of ways, they are necessary when users specify the goals they wish to achieve, rather than the actions they need to expose or perform (where the services fit well). Therefore usage of the agents may prove to be beneficial in non-trivial areas requiring rich interaction such as service matchmaking, contracting support, trust management etc.

Acknowledgements

The results presented here are partially funded by the European Commission under contracts IST-2003-01945 and IST-2005-027169 through the projects TrustCoM [39] and PANDA [40]. The authors would like to thank members of the partner organisations working in these projects: SAP, ETH Zurich, European Microsoft Innovation Centre, Imperial College London, SICS, Kings College London, University of Milano, University of Kent, BAE Systems, BT, IBM, Q-PLAN N.G., Altec S.A., Czech Technical University of Prague and others.

References

[1] Wombacher, A. et al. Matchmaking for Business Processes Based on Choreographies. 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04) pp. 359-368

[2] Vinoski, S. The Truth about Web Services. Web Services and Component Technologies. 2001. http://www.proinfo.com.cn/chinese/menu2/3/102501.ppt

[3] Web Services Architecture. W3C Working Group Note. 11 February 2004. http://www.w3.org/TR/ws-arch/

[4] Gortmaker, J., Janssen, M. & Wagenaar, R. SOBI Business Architectures and Process Orchestration: Technical Overview. Telematica Instituut, Enschede, The Netherlands. 2004.

[5] Merriam-Webster Online Dictionary, http://www.m-w.com

[6] Hofreiter, B. & Huemer, C. "Registering a Business Collaboration Model in Multiple Business Environments", Proceedings of the OTM Workshop on Modeling Inter-Organizational Systems (MIOS 2005) Larnaca, Cyprus, 2005

[7] Arenas, A., Djordjevic, I., Dimitrakos, et al. "Toward Web Services Profiles for Trust and Security in Virtual Organisations", Proc. 6th IFIP Working Conference on Virtual Enterprises (PRO-VE 2005), Valencia, Spain, 2005.

[8] A. Jøsang, C. Keser, and T. Dimitrakos. "Can We Manage Trust?" Proc. Third International Conference on Trust Management (iTrust), Rocquencourt, France, 2005.

[9] Dimitrakos, T., Djordjevic, I., Milosevic, Z., et al. "Contract Performance Assessment for Secure and Dynamic Virtual Collaborations", Proc. EDOC 2003, Brisbane, Australia, 2003.

[10] Morsel, A., "Metrics for the Internet Age: Quality of Experience and Quality of Business", Hewlett-Packard Labs Technical Report HPL-2001-179, 2001.

[11] Svirskas, A., Wilson, M., Arenas, et al. "Aspects of Trusted and Secure Business Oriented VO Management in Service Oriented Architectures", Workshop Proceedings of Seventh IEEE International Conference on E-Commerce Technology, IEEE Computer Society, Munich, Germany, 2005.
[12] Cherinka, R., Miller, R. & Smith, C. "Beyond Web Services: Towards On-Demand Complex Adaptive Environments", MITRE Technical paper, 2005.
[13] Dubray, J.J. WS-CDL - Choreography Description Language. http://www.ebpml.org/ws_-_cdl.htm
[14] Web Service Choreography Interface. W3C Note. 8 August 2002. http://www.w3.org/TR/wsci/
[15] ebXML Business Process Specification Schema. Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf
[16] Web Services Choreography Requirements. W3C Working Draft. 11 March 2004. http://www.w3.org/TR/ws-chor-reqs/
[17] Goland, Y.Y. A proposal for W3C Choreography Working Group Use Cases & Requirements. 16 May 2003. http://lists.w3.org/Archives/Public/www-archive/2003May/att-0029/chor.htm
[18] Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005. http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/
[19] Ross-Talbot, S. & Brown, G. Dancing in time with the services. WS-CDL. 1st Feb 2005 NY Java SIG, NYC, USA. http://www.javasig.com/Archive/lectures/JavaSIG-CDL-SRT.ppt
[20] Roman, D. Scicluna, J. & Feier, C. (eds.). Ontology-based Choreography and Orchestration of WSMO Services. WSMO Final Draft 1 March 2005. http://www.wsmo.org/TR/d14/v0.1/
[21] Feier, C. (ed.). WSMO Primer. WSMO Final Draft 01 April 2005. http://www.wsmo.org/TR/d3/d3.1/v0.1/
[22] Roman, D &Lausen, H. (eds.).Web Service Modeling Ontology – Standard (WSMO Standard). WSMO Working Draft 8 July 2004. http://www.wsmo.org/2004/d2/v03/
[23] Cimpian, E. & Mocan, A. WSMX Process Mediation Based on Choreographies. 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management (BPM 2005), September 2005, Nancy, France
[24] Dubray, J.J. ebXML. http://www.ebpml.org/ebxml.htm
[25] Enabling Electronic Business with ebXML. OASIS Whitepaper. 2000. http://www.ebxml.org/white_papers/whitepaper.htm
[26] ebXML Message Service Specification. Version 2.0 rev C. 21 February 2002. http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0rev_c.pdf
[27] ebXML Business Process Specification Schema v2.0. Working Draft 11, 5 April 2005 (Committee Draft candidate).
[28] WS Choreography WG conference call 15 March 2005. Minutes. http://www.w3.org/2002/ws/chor/5/03/15-minutes.html
[29] Schmidt, D. C. "Applying a Pattern Language to Develop Application-level Gateways," in Design Patterns in Communications (L. Rising, ed.), Cambridge University Press, 2000.
[30] Jennings N.R., & Bussmann S. Agent-based control systems. IEEE Control Systems Magazine 2003; 23:3, 61-74.
[31] Maximilien E.M., & Singh M.P., Multiagent System for Dynamic Web Services Selection. 2005
[32] Alonso G, Casati F, Kuno H, Machiraju V. Web Services Concepts, Architectures and Applications. Springer-Verlag, 2004
[33] Maximilien EM, Singh MP. "Toward web services interaction styles", In Proceedings of the 2005 IEEE International Conference on Services Computing (SCC'05), Orlando, Florida, USA, 11-15 July, 2005, pp 147-154.
[34] Maamar Z, Mostefaoui SK, Yahyaoui H. Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 2005; 17:5, 686 - 697.
[35] Ross-Talbot, S., ACM Queue - A Conversation with Steve Ross-Talbot on the role of choreographies and Pi-Calculus, http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=370&page=4
[36] Adams, H. et al. Best practices for Web services: Part 4, A Managed Public and Private Process Application Pattern Scenario. 2002. http://www-106.ibm.com/developerworks/library/ws-best4/?n-ws-12122
[37] Strader T.J., Lin F. & Shaw M.J. Information Structure for Electronic Virtual Organization Management. Decision Support Systems 1998; 23, 75-94.
[38] Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org
[39] Svirskas, A., Arenas, A., Wilson, M.D., Matthews, B. Secure and Trusted Virtual Organization Management. ERCIM News No. 63, October 2005, SPECIAL: Security and Trust Management. 2005
[40] Ignatiadis, I., Roberts, R. & Svirskas, A. Operational Support in Virtual Enterprises through Collaborative Process Automation", 1st International BIOPoM 2006 Conference, London, 2006