A Dynamic Orchestration Model for Future Internet Applications

Giuseppe Avellino[2], Mike Boniface[1], Barbara Cantalupo[2], Justin Ferris[1], Nikolaos Matskanis[1], Bill Mitchell[1], and Mike Surridge[1]

[1] University of Southampton IT Innovation Centre
2 Venture Road, Chilworth Science Park, Southampton, SO16 7NP
{mjb,nm,bm,ms}@it-innovation.soton.ac.uk
[2] Elsag Datamat spa, Via Laurentina 760, 00143, Rome, Italy
{giuseppe.avellino,barbara.cantalupo}@elsagdatamat.com

Abstract. Society and business are demanding systems that can securely and cost-effectively exploit opportunities presented by an Internet of Services. To achieve this goal a system must dynamically adapt to its environment and consider multiple and shifting stakeholder concerns such as application functionality, policies and business processes. In this paper we describe a dynamic orchestration model called the Virtual Infrastructure Model (VIM) which allows consumers to develop service-oriented systems that adapt to the needs of different business actors. It is based on the idea that adaptive workflow and dynamic binding to services can facilitate abstraction of both business processes and requisite interactions with the underlying infrastructure. Key requirements for federated orchestration are addressed including runtime service binding, secure and accountable dynamic procurement, infrastructure adaption, and separation of stakeholder concerns. The VIM is a fundamental component of the Next Generation Grid Architecture developed in the context of the EU funded NextGRID project.

Keywords: SOA dynamics, orchestration, lifecycle, workflow, business processes.

1 Introduction

After more than 10 years research and development of service-oriented systems, an economically viable Internet of Services have yet to materialise. Current software engineering theories, service specification and composition approaches assume software lifecycle models that significantly restrict the potential of service-oriented systems and the ability of such systems to support meaningful and dynamic social and economic relationships between communities and business partners.

Service-oriented systems are formed through the "recruitment" of services, possibly provided by different organizations, which are then orchestrated to achieve a desired objective. They cannot be subjected to conventional design "in advance", so developers are left to create parts of a system that must fit together in ways that cannot be anticipated until run-time. Future service-oriented systems will therefore need to be operated by business stakeholders rather than developed by engineers. To achieve this operational model new orchestration approaches are required that support multiple stakeholder interactions allowing them to manage the lifecycle of assets and interact with other stakeholders flexibly and dynamically whilst considering distributed policy and regulatory compliance.

In this paper we describe an orchestration framework to makes service-oriented systems adaptable to the needs of different business stakeholders. This framework has been termed the Virtual Infrastructure Model and has been designed and developed in the context of EU funded NextGRID project [1]. The architecture is based on the idea that adaptive workflows and dynamic binding to services can facilitate abstraction of both business processes and requisite interactions with underlying infrastructure separating functional system aspects from the business processes that govern service interactions. A prototype of the VIM has been implemented and reference scenarios in several application domains have been developed to validate effectiveness of the approach in real business contexts.

2 Dynamic Service-Oriented Systems

The ability to select and use services from a variety of independent sources and to integrate them into a system that delivers the functionality and performance desired is required for dynamic service-oriented systems. In any service-oriented architecture (SOA) the key functional components are services and systems. The fact that systems can be considered to be composed of services, which services themselves are provided by systems, is important. The recursive self-referential characteristic of SOAs is why they are so powerful. However, it also means that service-oriented systems can quickly become extremely complex concealing lower level structures from end-users.

The complexity is dramatically increased when systems are built from an Internet of Services that incorporates a multitude of federations between service consumers and providers. Business relationships are generally codified in contracts making all relevant details explicit such as defining what is to be provided at what service level, relevant business practices and standards to be used, as well as pricing and penalties for failing to meet the specified conditions. In an Internet of Services, these terms are expressed in Service Level Agreements (SLAs) that identify the business context for relationships between systems and services and determine many of the technical policies that govern the interactions.

Federation is established in service-oriented systems by introducing business processes that result in federation contexts (SLAs) that provide a link between *access to service* and the *management the service*. To achieve federation in a dynamic way each of these aspects need to specified separately allowing systems to be built independently of the business models for provision and procurement of services from which they are composed. By introducing a separation of concerns, multiple stakeholder objectives can be supported and used to govern systems as they are operated. In addition, service providers will host different infrastructures with different business policies, and this may be true even when they offer the same service functionality. When consumers need their system to achieve a functional goal services need to be selected at runtime from multiple, sometimes competing, service providers. It is the dynamic orchestration of business relationships that supports the delivery of system functionality in a secure, trustworthy and accountable way that will provide an essential enabler to an economically viable Internet of Services.

3 The Workflow Landscape

Workflow is a critical technology for the orchestration of the interactions between systems and services. Workflow is important because it can be considered as the programming language for service-oriented systems and therefore has the potential to support process flexibility by soft-coding system behavior. In a SOA context, workflow is used to express a composition of services and there are several competing standards, initiatives and many more proprietary solutions.

The most widely used specifications for describing procedural workflows within businesses are XPDL [2] and ebXML [3], and the most widely used workflow specification with reference to SOAs is WSBPEL [4]. The focus of BPEL, and most business-oriented workflow languages, is control flow. However, extensive research on workflow control patterns has shown that all languages have limitations in terms of what can be easily expressed [5]. This insufficient expressivity and lack of rigorous semantics significantly limits their ability to support adaptation mechanisms and dynamics. Van de Aalst provides an extensive pattern comparison of workflow languages and implementations. Whilst the post-hoc evaluation of existing workflow languages against workflow patterns with well defined semantics is useful, it does not address the problem of inherent lack of rigorous machine-interpretable semantics within each workflow language.

The semantic web service community, on the other hand, is producing rigorous models and logics for the semantic description of Web Services. Several European projects inc. SEKT, DIP, SUPER, ASG are working together through the European Semantic Systems Initiative and have collaborated to develop the Web Service Modelling Ontology (WSMO) [6] and Web Service Modelling Language (WSML) [7]. Meanwhile work done by academia and industry through SWSI has resulted in the Semantic Web Service Framework (SWSF) [8], which has both a language (SWSL) [9], and an ontology (SWSO) [10] (based on OWL-S) that includes a process model. These languages and models make workflows more amenable to machine reasoning, making it easier to create abstract representations of processes and runtime binding to the services that incorporate the both functionality and QoS.

As far as we can ascertain, very few of the current approaches are considering the need to consider dynamic stakeholder concerns in workflow orchestration and as far as we can establish none have attempted to design and implement a complete architectural model addressing all the issues described in section 3.

4 Virtualised Infrastructure Model

The vision of the VIM is to provide a run-time adaptable infrastructure that meets the key requirements for dynamic systems operating in an Internet of Services, in particular:

- **Run-time bindings:** system workflows need not specify a binding of every task to a specific service, so that the bindings can be chosen at run-time.
- **Selective enactment:** a single service may provide multiple functions, and it must be possible to choose which is bound to an abstract task, supported by the service.
- **Workflow substitution:** some abstract tasks may be bound at run-time to more detailed workflows that can be inserted into the enactment at run-time. A common

example is substitutions with template business operations such as account and billing workflows.

- **Workflow prioritization**: Critical processes, which are either expensive in resources or define the result or the performance of the workflow, must have high priority in the evaluation order.

A key feature of the VIM's approach is the abstraction of business processes so developers do not have to encode business processes explicitly in their systems. This allows systems to remain functional even if a service provider wants to use a different business model or process (e.g. pay-as-you-go instead of subscription-based access to services). The result is a workflow enactment model with a corresponding workflow enactment engine that provides a way to dynamically assemble system functionality using an abstract application workflow specification as a starting point, and introducing business processes at run-time as specified by the service providers and consumers involved in executing the application.

The capability is achieved by combining adaptive semantic workflow, semantic discovery and service selection heuristics with supporting business and security services that govern functional services. System logic can be captured with abstract "application workflows" that include the functional constraints of the system. Service providers can publish workflows to describe the interactions and preconditions necessary for a consumer to use their service. During workflow execution, abstract tasks are resolved to concrete implementations including business process steps through a process that includes discovery, selection and rewiring, before execution.

4.1 Workflow Enactment Model

The overall enactment model for the VIM is illustrated in Figure 1. At its core the VIM provides an ***Enactor*** that is based on "evaluate - apply" cycles, as used in functional

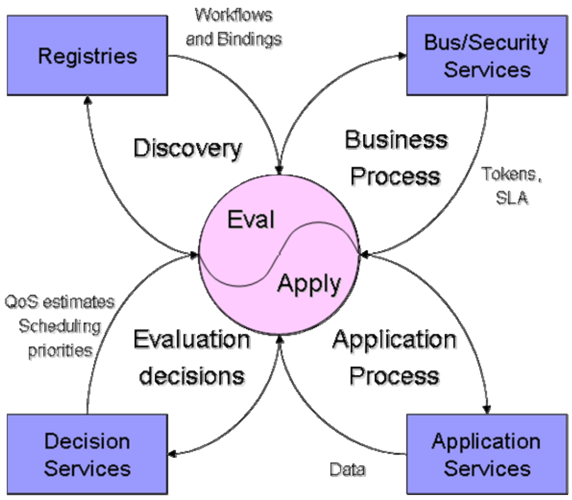

Fig. 1. VIM Enactment Model

programming. The aim of the evaluation is to replace abstract service descriptions with concrete services at runtime using components of the environment. Evaluation produces "concrete" processes that are either concrete ***Application Services*** or sub-workflows, which may contain abstract processes that also need to be evaluated. The apply phase that follows, executes the realized concrete processes. The evaluation algorithm includes four phases: prioritisation, candidate discovery, federation context acquisition, and candidate selection.

The evaluation order of a set of abstract processes in a workflow is determined by both enactor evaluation policies and prioritisation. Evaluation policies dictate whether to perform lazy or eager evaluation of conditional expressions, or when and how to fully evaluate nested composite workflows. Prioritization assigns priority weights to the workflow graph. Abstract processes with highest priority are evaluated first. Abstract processes that share the same priority level are evaluated together. Prioritisation helps the enactor to locate problems with availability of bindings for the abstract processes on critical parts of the workflow (e.g. missing SLAs). It also allows the enactor to to optimize execution by considering dependencies and data/control constraints.

Once prioritisation is completed candidate bindings for abstract processes are discovered from one or more ***Registry Services*** within the consumer's organisation.

In order to execute a candidate a consumer may need to acquire a federation context from supporting ***Security and Business Services.*** For example, if a SLA cannot be found, the enactor will use negotiation to establish a new SLA. SLA negotiations may also be required if service discovery fails to find any candidates. The negotiation of new SLAs can then allow access to more services, and when the SLA is agreed these services are added to the candidates list.

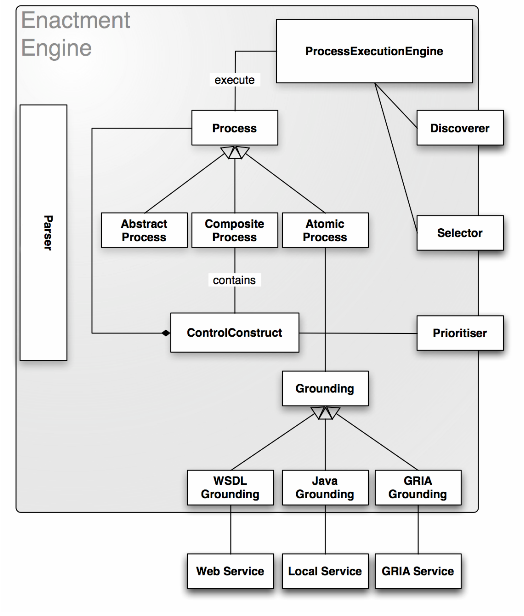

Fig. 2. Enactment Engine

Lastly the run-time binding of each abstract task to one of the candidates is determined using ***Decision services*** that apply selection heuristics and local organization policies. Selection operates across the whole workflow and may take account of co-location and other constraints. After selection of a candidate, replacement is made by rewiring the workflow and the ("Apply") phase is executes the task.

The workflow representation language that we adopted to represent workflows is OWL-WS. OWL-WS stands for "OWL for Workflows and Services" and is a workflow (and service) ontology fully based on OWL-S [11]. OWL-WS extends the OWL-S concept of Service to Abstract Process (an Atomic Process without implementation information), and uses the OWL-S concept of Composite Process for workflow modelling. In OWL-WS, Profile is available to any Process providing the ability to represent information at any level of the workflow composition. A more detailed description of the language is provided in [12]. A detailed model of the VIM can be seen in Figure 2.

4.2 Workflow Enactment Engine

The workflow enactment model has been implemented by integrating a range of service-oriented technologies. The ***Enactor*** is based on the Mindswap OWL-S API [13],which supports representation and enactment of OWL-S elements. Mindswap was extended to provide additional features to support OWL-WS extensions, and more complex and dynamic eval/apply execution semantics.

The evaluation order of an Abstract Process can be set both manually by the workflow author and automatically by the ***Prioritizer*** component. The Prioritizer uses QoS and historical information to assign evaluation priorities to those abstract processes that have not yet been prioritised. The ***Discoverer*** component looks for candidate bindings for an abstract process starting from its Profile. The Profile expresses constraints on the discovery process, effectively encapsulating a query that should be used to locate candidates. Discovery is performed by querying service registries that are located in the consumer's domain. This registry implementation is based on the Globus GT4 implementation of the WSRF-SG specification, and supports the XPath query language [14].

Federation context acquisition is implemented using the ***SLA Discovery*** and ***Broker*** components. The SLA Discovery component retrieves SLAs from a SLA registry. Many concrete services can only be executed under an agreement with the service provider, so an SLA reference is essential for the execution of these services. Negotiation of new SLAs with service providers is performed by means of the Broker component. The broker uses service provider registries to look for advertised services that fulfill consumer requirements. Once these have been decided the negotiation process takes place in order to establish a new SLA. The SLA that is produced is then registered to the SLA Registry followed by an update to the application service registry to register new service functionality that has been procured. The current Broker implementation consists of five components: the Matchmaker, the Negotiator, the Reconciler, the Template Retriever, and the Deal Closer and is described in detail in [15].

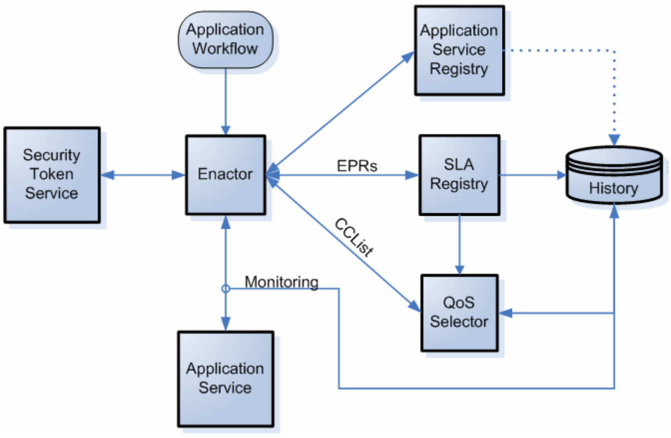

Fig. 3. VIM Components

The final selection of service bindings is performed by a ***Selector*** component. The selection process involves choosing a single candidate for each Abstract Process from the set of candidates found in the discovery phase. The selector implements an algorithm that takes into account criteria taken from an SLA, historical data and service parameters. User hints may also influence the selection. In order to determine the best from the available candidates for each step the selector takes into account services selected in other nodes of the workflow. The enactor gathers information of the service performance under a SLA, and can then be submitted to a Quality of Experience analysis service if the user so chooses. The QoE analysis can then also be used to produce criteria for the Selector component, based on previous experience of the candidate service providers [16].

Apart from the evaluation components the enactor uses Groundings to infrastructure implementations. These Groundings encapsulate the information required to construct and send appropriate messages to services and other executable components that are external to the Enactor. The WSDL and Java groundings enable Web Service and local service invocations. The GRIA grounding supports services hosted by the GRIA middleware [17], while the NextGRID grounding provides similar functionality but supports NextGRID specifications for SLAs and exchanged messages.

5 Real Context Experiments

The VIM architecture has been verified by architectural experiments and used in reference applications within the NextGRID project [18]. These applications include:

- Digital Media (DM): This application uses workflows that consume Rendering services for a for a television advertising company.
- Electronic Data Record (EDR). This application uses Grid services for a telecommunications company.

In the following paragraph we chose to analyse the Digital Media experiment. In the DM scenario, users want to run video rendering application workflows that have been written with the OWL-WS workflow-authoring tool by the application system experts. These workflows are abstract. The users have control over which workflow to use, over its input data and over the parameters for the execution through a web-portal. Parameters can specify preferences on price, availability, required time or other business factors. The application workflow in this scenario is shown in Figure 4:

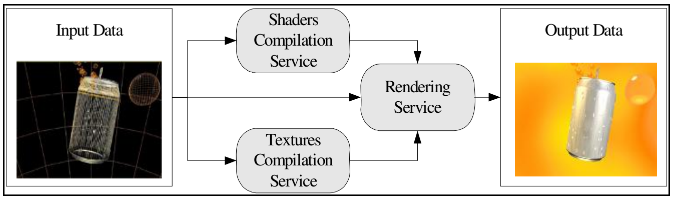

Fig. 4. 3D Video Rendering Scenario

Each of the abstract processes in this workflow is resolved by discovery and selection to a concrete application process with the SLA EPR information. The discovery performed in an order defined by the prioritiser implementation and the selection is taking into account the SLA terms, user preferences and Quality of Experience analysis results. The selection is done through out the workflow to take into account co-allocation issues according to the selector implementation.

This experiment demonstrated the ability of the system, through its user interface, to setup the environment of VIM infrastructure and enact abstract workflows of the video rendering application with different inputs and parameters. By changing the QoE parameters, users influenced selection and led to different concrete workflows that had different business models. In any case the abstract processes were evaluated by the VIM and concrete workflows with business management rules and policies were successfully enacted using NextGRID compliant application services.

6 Conclusions and Future Outlook

In this paper we presented an orchestration architecture for Future Internet applications based on a dynamic and adaptive workflow model. The architecture addresses the key requirements for service-oriented systems operating in an Internet of Services (runtime binding to services, secure and accountable dynamic procurement, infrastructure adaption, and separation of stakeholder concerns). A prototype workflow engine with related components has been developed and validated in significant business applications demonstrating how the lifecycle of system functionality and business processes governing underlying services can be separated.

The current implementation is limited to adapting consumer systems to service provider business processes by injecting these processes into application workflows at runtime. As we move towards an Internet of Services, consumers require systems that deal with the increased complexity and allow them to assess and mitigate threats in a more open world. To deal with these issues multiple business stakeholders (operations, finance, legal, quality, and marketing) will govern interactions and will work together to achieve an overall business objective. Effectively the atomic view of a consumer or service provider business process will be insufficient as multiple consumers will need to orchestrate their perspectives in goal-oriented event driven approach. This will require more fine-grained adaptive workflows to manage the lifecycle of different aspects of systems and services.

The creation and governance of applications of service-oriented infrastructures must become much easier for all stakeholders as the diversity and scale of assets dramatically increases, especially for applications that span multiple administrative domains. The VIM orchestration model introduces dynamics into service-oriented systems in a way that could not be previously achieved. Future work will continue to focus on orchestrating federations and will examine how the VIM model can be enhanced by applying functional programming and process algebra approaches to dynamic service composition and agent-based functionality in decision services.

References

1. Next Generation GRIDs Expert Group Report 3, Future for European Grids: GRIDs and Service Oriented Knowledge Utilities (January 2006)
2. Workflow Management Coalition Workflow Standard, XML Process Definition Language (XPDL), Document Number WFMC-TC-1205 FINAL: Version 2.0, October 3 (2005)
3. OASIS standard v2.0.4, ebXML Business Process Specification Schema Technical Specification v2.0.4 (December 2006)
4. Alves, A., et al. (eds.): Web Services Business Process Execution Language Version 2.0, OASIS Committee Specification (January 2007)
5. Workflow Control-Flow Patterns A Revised View. Nick Russell, Arthur H.M. ter Hofstede (BPM Group, Queensland University of Technology) and Wil M.P. van der Aalst, Nataliya Mulyar (Department of Technology Management, Eindhoven University of Technology)
6. `http://www.wsmo.org/`
7. `http://www.wsmo.org/wsml/`
8. `http://www.w3.org/Submission/SWSF/`
9. `http://www.w3.org/Submission/SWSF-SWSL/`
10. `http://www.daml.org/services/swsf/1.0/swso/`
11. Martin, D. (ed.): OWL-S: Semantic Markup for Web Services, W3C Member submission (November 2004)
12. Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N., Surridge, M.: OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. In: 1st Int. Conf. on e-Science and Grid Computing (2005)
13. See Mindswap OWL-S API project, `http://www.mindswap.org/2004/owl-s/api/`
14. Hasselmeyer, P.: On Service Discovery Process Types. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 144–156. Springer, Heidelberg (2005)
15. Hasselmeyer, P., et al.: Towards Autonomous Brokered SLA Negotiation. In: Paul, Cunningham, M. (eds.) Exploiting the Knowledge Economy - Issues, Applications, Case Studies, vol. 3. IOS Press, Amsterdam (2006)
16. McKee, P., Taylor, S.J., Surridge, M., Lowe, R., Ragusa, C.: Strategies for the Service Marketplace. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 58–70. Springer, Heidelberg (2007)
17. GRIA Middleware for Service Oriented Collaborations for Industry and Commerce, `http://www.gria.org/`
18. NextGRID application fliers, Digital Media application, `http://www.nextgrid.org/download/flyers/NextGRID%20Digital%20Media%20Flyer.pdf`, Electronic Data Records application, `http://www.nextgrid.org/download/flyers/NextGRID%20Digital%20Media%20Flyer.pdf`