A Model-Driven Architecture for Electronic Service Management Systems

Giacomo Piccinelli[1], Wolfgang Emmerich[1], Scott Lane Williams[2], and Mary Stearns[2]

[1] Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK
{G.Piccinelli, W.Emmerich}@cs.ucl.ac.uk
[2] HP Software and Solutions Operation, Pruneridge Avenue, Cupertino, CA 95014, USA
{scott_l_williams,mary_stearns}@hp.com

Abstract. Mainly on the wake of the Web Service initiative, electronic services are emerging as a reference model for business information technology systems. Individual applications retain core functions and technology base, but integration becomes crucial. A business service derives from the coordination of different business capabilities. The related electronic service derives from the integration of the different applications sustaining such capabilities. The effective realisation of an electronic service requires explicit modelling and active management of the relations between business capabilities and technical infrastructure. In this paper, we propose the notion of Electronic Service Management System (ESMS) as a framework for modelling and implementing electronic services. The notion of ESMS is substantiated by a workflow-oriented architecture, which we mainly derive from the experience of HP Service Composer and the DySCo (Dynamic Service Composer) research prototype. The architecture is defined in accordance with the OMG's Model-driven Architecture (MDA) principles.

1 Introduction

Electronic services are based on the convergence of the technical and the business notions of service [14,20]. On the technical side, standard and technology initiatives such as Web Services [3] provide a new delivery channel for bodies of knowledge such as the RM-ODP (Reference Model for Open Distributed Processing) [10]. On the business side, services provide an established unit of modularisation for business capabilities [4].

The challenge on the business side is to adapt business infrastructure and models to service-oriented principles. For example, re-engineering internal assets and functions as services. The challenge on the technical side is to provide a framework for electronic services that is both comprehensive and accessible. The realisation of an electronic service requires explicit modelling and active management of the relations between business capabilities and technical infrastructure.

M.E. Orlowska et al. (Eds.): ICSOC 2003, LNCS 2910, pp. 241–255, 2003.

Models must support abstractions at different levels, and the links between levels must be explicitly formalised.

In this paper, we propose the notion of Electronic Service Management System (ESMS) as a conceptual and technical framework for electronic services (Section 2). The framework (Section 5) includes a structural and operational definition of electronic service, the notion of service composition, and a blueprint for service implementation. The framework is defined in accordance with OMG's MDA (Model-driven Architecture) [6] principles, and draws upon OMG's EDOC (Enterprise Distributed Object Computing) [16] specification. The concepts proposed derive mainly from the experience of HP Service Composer (Sections 3 and 4) and the DySCo (Dynamic Service Composer) [18] research prototype. DySCo also provided an initial validation platform for electronic service modelling and implementation (Section 6). Related work is discussed in Section 7. Conclusions and future directions are discussed in Section 8.

2 Electronic Service Management

The definition of electronic service (service for shorthand) adopted for our work is that of electronic virtualisation of a business service [14]. The notion of electronic service inherits richness as well as complexity from the business notion of service. In addition, the electronic dimension introduces new issues in terms of both service content and provision.

The content of a service refers to the core capabilities enabled by the service. For example, the content of a freight service refers to the capability of moving goods from one place to the other. Provision refers to the business channel [8] between the provider and the consumer of a service. In the example, provision covers selection, product offer, pricing, and interaction processes that the freight company applies to the users. Content and provision are complementary aspects of a service. On the one side, the provision logic depends on the capabilities that the provider can support. On the other side, the capabilities made available to consumers depend on the provision logic adopted by the provider. In the example, the option of delivery tracking might be made available only to selected users. The example is based on previous research in the freight domain [13], and will be used throughout the paper.

The notion of Electronic Service Management System (ESMS) that we propose is centred on a framework for the representation of the operational logic of an electronic service, the representation of the resources involved in the content and provision of the service, and the active coordination of such resources in accordance with the operational logic of the service. An ESMS does provide a conceptual and technical infrastructure for the development and management of electronic services. For example, the ESMS can model and access the order-management system of the freight company. An ESMS does not address the business definition and engineering of the services. In the example, the ESMS would not influence the design of a new transport service. Also, an ESMS contributes to the coordination of resources involved in the content and provision of

electronic services. For example, the ESMS can manage the interaction between the inventory and the order-management systems. With the exception of coordination facilities, an ESMS does not contribute new resources to be used within a service. The ultimate goal of an ESMS is to bring together resources that underpin an electronic service. The notions of workflow [5,7] and composition [14,16] are fundamental, both from a conceptual and technical perspective.

The rationale for an ESMS derives from current practices for the creation and management of electronic services (see Section 3). In the general case, service providers control resources of different types that are used in different combinations in order to produce different types of service. Different services can depend on the same resources, and the execution of one service can affect the execution of other services. For example, the same truck might be used to dispatch both perishable and non-perishable goods (compatibly with the type of package). Usage conflicts can easily occur. Moreover, different services can be related in different ways. Some services can provide complementary capabilities. In the example, a customer might need to move both perishable and non-perishable goods. Other services can instead provide alternatives for the same capability. For example, a repackaging service in combination with the transport service for non-perishable goods could provide an alternative to the transport service for perishable goods. The business knowledge developed for traditional service management must be reflected in the electronic version of a service. Presenting a provider with a coherent view of resource base, service offer, and the interdependencies between resources and services, ESMSs shorten the gap between business design and technical implementation of electronic services.

The choice of OMG's MDA as modelling technique for ESMSs reflects the need for a multi-stage approach in the development of systems for which integration is a crucial issue. Beyond communication-level protocols, the realisation of an electronic service depends upon the close integration of different applications and systems. Moreover, the dynamics of change in the resource base for a service makes the ability to adapt a fundamental requirement. The MDA provides the base for effective system integration and reengineering.

3 Case Study

The notion of electronic service management system has found inspiration as well as validation in the HP Service Composer (HPSC) [9]. The technology framework of the HPSC includes tools and infrastructure components for workflow and data modelling, workflow execution and management, back-end integration, and Web-Service lifecycle management. Most importantly, the HPSC includes a methodology for the definition and development of electronic services. The essence of the methodology is captured in the following steps:

1. **Define Public Business Processes.** The developer defines the public workflow that clients will use to interact with the service. The developer either selects an existing process definition, or defines new ones.

2. **Program Web Service Interfaces.** The developer generates the Web Services Description Language (WSDL) files, which describe the Web Services associated to the process of step one.
3. **Generate Business Objects and Data.** The developer generates or creates connections to the business objects and data that support the service.
4. **Define Internal Business Processes.** The developer defines the internal workflow specifying the operational logic for the service. For pre-existing workflows, the developer builds access points to relevant process nodes.
5. **Map Public Interfaces.** The public interfaces defined in steps one and two are mapped to back-end logic from steps three and four. As an example, a WSDL interface might be mapped to a backend component for its concrete implementation.
6. **Package the Service.** The various components and descriptor files that make up the service are combined into a deployment unit. The deployment unit can vary depending on the target platforms.
7. **Deploy the Service.** The service is deployed onto the various components of the runtime platform (e.g. application server, workflow engine, ERP– enterprise resource panning – system, Web Service infrastructure).
8. **Advertise the Services.** Once a service has been deployed, it can be offered to clients. For example, entries for the related Web Services can be added to a UDDI repository.
9. **Monitor Running Services.** Graphical tools should provide an end-to-end view of the service at instance level or as aggregates.

Aligned with industry trends such as ebXML [8] and technology trends such as Web Services, HPSC is representative of the state-of-the-art in commercial systems. As all commercial products, HPSC balances innovation with concrete and immediate applicability. From the study of the HPSC case, we derived general requirements (Section 4) for an electronic service management system. In particular, the methodology for electronic service development gave us indications about the facilities expected by an ESMS. Most important, some of the architectural choices in the HPSC provided fundamental indication on the concept of platform for an ESMS (Section 5).

4 System Requirements

The most important cluster of requirements derived from the experience with HP Service Composer concerns the granularity of business resources. An ESMS must represent the business capabilities enabled by software systems. Using RM-ODP [10] terminology, an ESMS must expose and leverage the business view of the different systems and applications sustaining an electronic service. For example, the ESMS for a freight company should expose to the service designer the customer interaction logic for the order management system. A homogeneous and coherent view on business resources is a prerequisite for the engineering of an electronic service.

Different resources must be integrated at operational, as well as communication level. Hence, the operational logic of a business resource should be explicitly modelled and actively managed. In the previous example, the order management system must be integrated with the payment system. This implies that the two systems communicate, but also that communication occurs according to a mutually acceptable process. Workflow emerges as a widely accepted model for the representation of the operational logic of business resources. Workflow also emerges as a reference model for the representation of the operational logic of an overall electronic service. In the case of workflow as well as other models and technologies, standard-based solutions are fundamental.

The distinction between resources and services is quite important. The vision of resources and services becoming indistinguishable entities sets a long-term objective. Still, an ESMS must address the peculiarities of resources as well as services (see Steps 1-5, Section 3). Accessibility, availability, control, and cost are only some of the aspects to which the distinction applies. While services and resources should be treated as different entities, an ESMS must address the inter-category relations. Resources sustain services. In the freight example, a lorry sustains the transport service. At the same time, services sustain higher-level resources. In the example, insurance services sustain the operation of the lorry. An ESMS must also address intra-category relations for services as well as resources. Services can be composed to create new services. For example, composing pure transport with packaging and progress notification can produce a new type of delivery service. Resources may also be used in composition. For example, different trailers need to be matched with a suitable truck. An ESMS must explicitly mange the various relations for services and resources.

5 MDA Model for ESMSs

The conceptual and technical framework we propose for an ESMS is defined based on OMG's MDA (Model-Driven Architecture) [6]. Both the EDOC (Enterprise Distributed Object Computing) [16] and the RM-ODP (Reference Model for Open Distributed Processing) [10] specifications constitute the conceptual foundational for ESMSs. In addition, the EDOC specification provided the template for the definition and formalisation of the ESMS model. In line with the EDOC approach, the purpose of an ESMS model is to provide a reference framework to developers of ESMS systems. In particular, the definition of a UML Profile provides immediate support for ESMS modelling and design.

After an outline of structural and operational aspects of the ESMS architecture (Section 5.1), we introduce a set of key concepts for an ESMS (Section 5.2). A metamodel [1] for ESMSs is presented in Section 5.3. The formal semantics of the proposed metamodel is discussed in Section 5.4.

5.1 Outline of the ESMS Architecture

The architectural model we propose for electronic services and related management systems is based on the definition of *business asset*. A business asset is

defined as the composition of one or more business resources and services into a self-contained operational unit. For example, the composition of a lorry, a driver, and an insurance service can constitute a single asset capable of moving goods between different locations.

At the next level of aggregation, the composition and coordination of assets provide the foundation for *business capabilities*. A business capability is defined as the composition of one or more business assets in accordance with an explicitly defined business process. For example, the composition of the transportation asset with assets for packaging, loading and unloading constitute the base for a delivery capability. A business process coordinating the various assets completes the capability. One asset can contribute to the realisation of multiple capabilities. Also, different capabilities can interoperate. The interoperation logic for a *group* of capabilities emerges from the business processes of the individual capabilities. While the relation between capabilities and assets has a master-slave connotation, the interoperation between capabilities is based on a peer-to-peer approach. For example, capabilities for delivery, insurance and billing may interoperate for the fulfilment of a customer request. From a business process perspective, the difference is between the definition of a process coordinating multiple resources (capability) and the federation of multiple processes (capability group). The boundaries between assets and capabilities can vary between different business domains, as well as between different organisations in the same domain. Variability can occur also in different parts of an individual organisation.

In addition to the operational interdependencies captured by groups, business capabilities can functionally complement each other. A *cluster* captures the relation between a set of business capabilities and a specific business function or segment. For example, transport and handling capabilities for perishable goods can be aggregated into one or more clusters in the segment for perishable freights. Similarly, capabilities for accounting and payment management can be aggregated into a cluster for finance. One capability should contribute to at least one cluster. Most important, one capability can contribute to more than one cluster. Communication handling is an example of capability that is required in different business functions and segments. Relations and dependencies between clusters derive mainly from the relations and the dependencies between the capabilities contained in individual clusters. The new dimension introduced at cluster level is the distinction between content and provision (ref. Section 2). In the previous example, the cluster for perishable freights relates to the content of a fright service. The cluster for finance relates to the provision of the service.

An *electronic service* emerges from the interoperation between a specific selection of content and provision capabilities. The selection should include content-related as well as provision-related capabilities. Aspects of a service such as information and resource requirements, operational characteristics, and user interaction derive from the capabilities involved as well as the way such capabilities are set to interoperate. For example, the address for a delivery may be obtained through the interaction with a profile management capability. Alternatively, the information may be obtained by direct interaction with the user.

The realisation of a service S can be based on other services $\{S1 \ldots Sn\}$. The services Si are independent services that act as *sub-services* for S. The different Si can be provided by the organisation that implements S, or by third parties. Independently from the provider of Si, the connection with S is handled as if S were a standard user. Similarly to clusters for capabilities, services can be aggregated in *service packs*. A service pack includes services that complement each other in terms of business content. As for sub-services, a service pack can include services provided by third parties. The rationale is to present users with comprehensive solutions that leverage the product (services) of an organisation. A key difference between a composed service and a service pack lays in the responsibility for the end result. In the case of a composed service, the provider is responsible for the overall result delivered to the user. In the case of a service pack, individual providers are responsible for the results of individual services. Still, the user is responsible for the result of their combined usage.

A *service offer* is the user view on an electronic service. Service offers include elements such as price, contractual terms and conditions, interaction processes, and information on business content. Different users may be presented with different views on a service, hence with different service offers. Business roles [5] are central to business processes, hence to business capabilities and services. In concrete terms, the user view on a service derives from the *business roles* assigned to such user.

5.2 Basic Concepts for an ESMS

The concept of platform is fundamental for both the modelling and the realisation of an ESMS. Organisations have in place information systems that provide both modelling abstractions and technology infrastructure for electronic services. An ESMS must leverage abstractions of business resources and capabilities already present within an organisation. Also, the realisation of an ESMS must build on top of the technology already present in an organisation. The reference platform for an ESMS includes elements such as ERP (Enterprise Resource Planning) systems, database management systems, business process management systems, and domain-specific as well as function-specific vertical applications. Integration middleware (e.g. Web Services) already provide uniform access to the platform elements. An ESMS provides uniform abstractions and application-level integration for such elements.

Assets can be abstracted using a basic object-oriented model (e.g. the model used by Web Services [3]). Taking a progress-tracking system as an example, the system can be modelled as a Web Service that takes an order number in input and returns the number of hours before the expected delivery. The majority of existing information systems expose interfaces based on some form of object-oriented model. Further assumptions would be useful (e.g. explicit modelling of interaction flows), but in most cases unrealistic. Capabilities can be abstracted using a workflow-oriented model (e.g. the model proposed by the Workflow Management Coalition [5]). Workflow is an established formalism for the definition

of business processes. Moreover, most organisations use workflow management technology.

The rationale for groups and clusters is to support the modularisation of the overall business infrastructure underpinning a service. An electronic service is defined and engineered on top of the business infrastructure provided by capabilities. Explicit modelling and active management of the relations between capabilities provide the base for modelling and managing services. In general, orchestration-oriented coordination of all the capabilities intervening in the realisation of a service is difficult to achieve. For example, explicit workflows covering end-to-end freight services would reach levels of complexity not acceptable in practice. Still, orchestration is possible and necessary for specific aspects of a service. In the freight delivery example, a workflow may coordinate order management, billing, and credit card handling capabilities to manage payments. A second workflow may coordinate the actual delivery of the goods. A third workflow may coordinate the handling of customer queries. The overall service emerges as a loose form of aggregation of such workflows. From a modelling perspective, coordination is a specific capability that can be grouped with the capabilities to be coordinated.

An ESMS requires an information base to manage all the information relevant to the operation of services. The concept of information base mainly encompasses uniform data models and access. Uniform data models and formats enable information sharing between capabilities. Uniform access enables source transparency. The realisation of an information base can imply proxy access to existing data sources, as well as database facilities. The information base can be leveraged for the realisation of loose forms of coordination between capabilities.

5.3 The ESMS Metamodel

Figure 1 captures the metamodel for an ESMS. The semantics for most of the elements in the metamodel can be derived from the discussion in the previous sections (Section 5.1 and 5.2 in particular). In addition, a formal semantics for the relation between capabilities, clusters, and services is presented in the following Section 5.4. For reasons of space, in this section we concentrate on properties, roles, rights, and responsibilities.

Properties capture different types of meta-information about capabilities. Such meta-information mainly refers to functional and non-functional requirements for a capability. For example, a property for a negotiation capability is to be usable only with a certain type of customers. A different type of property for the same capability may be the need to keep certified logs for all communications. A similar layer is defined for clusters. An example of cluster-level property is the response time to external customer requests. The relations between properties for clusters and capabilities are modelled explicitly. In particular, the emphasis is on the consistency between properties at different levels. For example, the response time specified for the cluster cannot be lower than the response time of the slowest capability.

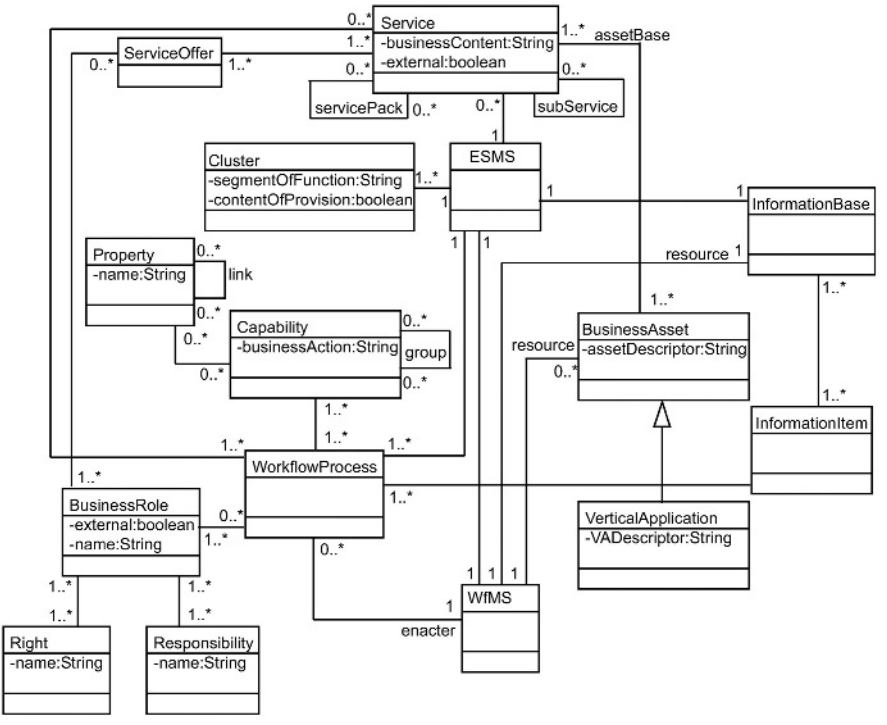

Fig. 1. Electronic Service Management System (ESMS) – Metamodel.

Rights and responsibilities are fundamental aspects for the notion of services. In an ESMS, centres of rights and responsibilities are modelled using a notion of role similar to the one used in RM-ODP [10]. The relation between roles, rights, and responsibility is subject to change over time. Such relation can be described by the following function:

$$map : Ro* \times S \rightarrow Ri* \times Re* \tag{1}$$

Given a set of roles (Ro), a set of rights (Ri), and a set of responsibilities (Re), the function connects roles to sets of rights and responsibilities. In the formula, $*$ indicates the power set of a set. S captures the notion of state, and the possibility of dynamic adaptation in the mapping. Ultimately, roles are mapped to software applications (e.g. modules of an ERP system). Rights and responsibilities are used to manage the activity of such applications (e.g. validate message flows).

5.4 Formal Semantics

A semantic characterisation of the ESMS metamodel is defined in a formal way by the Labelled Transition System (LTS) [15, 19] partially reproduced in Figure

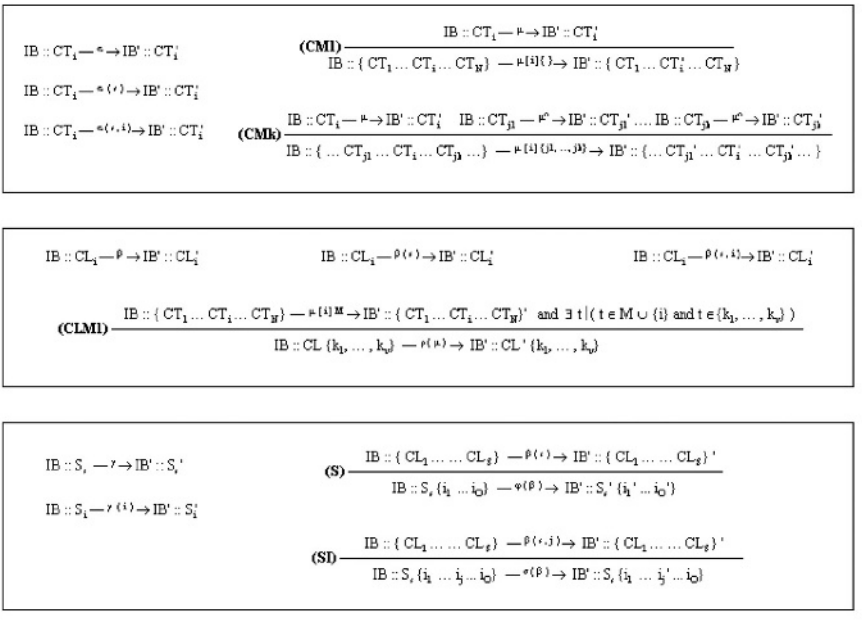

Fig. 2. Extract form the definition of the LTS of the ESMS. From the top down: capabilities, clusters, and services.

2. The LTS formalism provides an operational view of the ESMS, and a precise definition of the behaviour expected at different levels within the system. The focus is on the relations between capabilities, clusters, and services as described in Section 5.1.

As described in the ESMS model, a capability can be related to more than one cluster. The transition rules CM1 and CMk (Figure 2) indicate that capabilities can evolve either individually ($CM1$) or cooperatively (CMk). The evolution of a capability (from C to C') is associated with tangible results (μ). Different results can have different impact on services and service instances. Some results do not affect any specific service (α). Some results affect a service as a whole ($\alpha(s)$). Some results affect specific service instances ($\alpha(s,i)$). The results achieved cooperatively ($m[i]\{j1 \dots jk\}$) are annotated with indications of the capabilities involved ($j1 \dots jk$), and of the coordinating capability ($[i]$).

Clusters can evolve autonomously, but most commonly a cluster evolves as a consequence of the evolution of at least one of the capabilities it depends upon ($CLM1$). In the latter case, the result of the cluster evolution derives from the result of the related capabilities ($\rho(\mu)$). Multiple clusters can evolve as a consequence of the same evolution steps at capability level. As for the capabilities, the results produced at cluster level can have different degrees of impact on services and service instances (β, $\beta(s)$, $\beta(s,i)$). The evolution of services (S) and service

instances (SI) is directly related to the evolution of the clusters, which implies a degree of indirection with respect to specific capabilities.

The benefits of formal semantics and the use of LTS for describing an ESMS metamodel are in terms of simulation and analysis made available by tools such as the LTSA (LTS Analyser) [15]. Initial experiments indicate that the tool is particularly useful for scenario-based validation of the metamodel. Behavioural modelling and analysis validate the adequacy of the structural aspects of an ESMS.

5.5 UML Profile

In-line with the methodology defined for the MDA, Tables 1 and 2 outline the UML profile corresponding to the ESMS metamodel. The immediate benefit of the profile is that standard UML modelling tools can by used to support the design ESMSs. In particular, tools can enforce the compliance of specific ESMS models with the overall metamodel defined in the previous sections. As an example of the structural constraints included in the profile, the following OCL (Object Constrain Language) rule captures the need for a property to be associated with a capability or a cluster:

```
context Property inv:
-- a Property must be related to at least one Cluster or Capability
self.
association.
association.
connection.
participant
->excluding(self).stereotype
->exist(name = "stripe" or name="capability")
```

The profile is defined as a single package that mainly extends (¡¡access¿¿) the package for the Core metamodel in the UML Foundation. The current version of the profile addresses the modelling of structural aspects of an ESMS. Enabling UML-based modelling of the behavioural aspects of an ESMS constitutes ongoing work.

6 Proof of Concept

The definition and characteristics of the ESMS model derive substantially from the experience of HP Service Composer. The definition takes into account both the methodology for solution development (Section 3) and the associated toolset. UML notation is used in the HPSC to enforce the separation between platform-dependent and platform-independent models of an electronic service. Workflow notation and technology is used to model and manage the business logic of a service. The HPSC framework can be directly used to refine a platform-independent model compliant with the ESMS metamodel into a more detailed

Table 1. Stereotypes for ElectronicServiceManagementSystem (UML notation: Class Diagram)

Metamodel element	Stereotype	Base Class	Parent	Tags
Business-Asset	BusinessAsset	Class	N/A	assetDescriptor
Business-Role	BusinessRole	Class	N/A	name external
Capability	Capability	Class	N/A	businessAction
InformationBase	InformationBase	Class	N/A	
InformationItem	InformationItem	Class	N/A	
Property	Property	Class	N/A	name
Right	Right	Class	N/A	name
Responsibility	Responsibility	Class	N/A	name
Service	Service	Class	N/A	external businessContent
ServiceOffer	ServiceOffer	Class	N/A	
ESMS	ESMS	Class	N/A	
Cluster	Cluster	Class	N/A	businessFunction
VerticalApplication	VerticalApplication	Class	BusinessAsset	VADescriptor
WfMS	WfMS	Class	N/A	
WorkflowProcess	WorkflowProcess	Class	N/A	

Table 2. Tagged Values for ElectronicServiceManagementSystem

Metamodel attribute	Tag	Stereotype	Type	Mult.	Default
assetDescriptor	assetDescriptor	BusinessAsset	String	1	
businessAction	businessAction	Capability	String	1 . . . *	
businssContent	businssContent	Service	String	1 . . . *	
contentOrProvision	contentOrProvision	Cluster	Boolean	1	
external	external	BusinessRole	Boolean	1	false
external	external	Service	Boolean	1	true
name	name	BusinessRole	String	1	
name	name	Property	String	1	
name	name	Right	String	1	
name	name	Responsibility	String	1	
segmentOrFunction	segmentOrFunction	Cluster	String	1	
VADescriptor	VADescriptor	VerticalApplication	String	1	

platform-independent model. The HPSC can also be used in the generation of platform-dependent models for an ESMS.

The ESMS model is also closely related to the DySCo (Dynamic Service Composer) [18] research prototype. DySCo is the result of a two-year project involving University College London (UK), the University of St. Petersburg (Russia), the University of Ferrara (Italy), the University of Hamburg (Germany), and Hewlett-Packard (UK and USA). The objective of DySCo was the development of a conceptual and technology framework for the dynamic composition of electronic services. While lacking direct support for UML, DySCo provides modelling facilities for workflows and a homogeneous execution platform for an ESMS.

The ESMS model is currently being used in the context of the EGSO (European Grid for Solar Observations) [2] project. The model-driven approach to the architecture of the service provision part of the EGSO grid is expected to address the need to integrate services based on different provision models and execution platforms. Each service provider in the EGSO grid will be equipped with an ESMS. In addition, a specific ESMS federates and manages the service provisioning capabilities of the overall EGSO grid.

7 Related Work

The Enterprise Collaboration Architecture (ECA) defined in the OMG's EDOC specification [16] provides a comprehensive framework for the modelling of enterprise systems. The ESMS architecture can be considered at the same time a vertical extension and an instance of the ECA. On the one side, an ESMS is an enterprise system that can be designed based on the ECA. On the other side, an ESMS has peculiarities that are not explicitly addressed by the ECA. With reference to the conceptual framework for the MDA [6], we envision a series of refinements (PIM PIM) before reaching the level of detail at which the ECA can be effectively used. Similar considerations apply for the Reference Model for Open Distributed Processing (RM-ODP) [10], which is also closely related with the ECA.

Most technology and conceptual frameworks for electronic services [12] focus on Web-Service-based automation of the front-end of individual services. Web Services [3,4] constitute the reference model for access to and basic orchestration of business resources. We envision Web Services playing a fundamental role in the realisation of an ESMS. Still, a more comprehensive approach is needed for the realisation and operation of business-level services. An example of the issues involved in the realisation of business-level service is HiServ's Business Port [11]. FRESCO (Foundational Research on Service Composition) [17] provides an example of second-generation framework for electronic service management. The focus of FRESCO is on the provision aspects of services.

8 Conclusions

An ESMS (Electronic Service Management System) provides the conceptual and technology framework for the aggregation and coordination of business resources towards the realisation of a complete electronic service. In particular, the realisation and operation of a service requires close integration between different systems. A model-driven approach to ESMSs helps tackling the integration issue at multiple levels. In this paper, we propose a general architecture for ESMSs. The architecture derives from the specific experience of HP Service Composer, but it is also closely related to concepts in OMG's EDOC (Enterprise Distributed Object Computing) specification and the RM-ODP (Reference Model for Open Distributed Processing). The architecture is captured as an UML-based metamodel for which a UML profile is also defined. The architectural model proposed

has been applied to the DySCo (Dynamic Service Composer) research prototype, and it is currently been used in the EGSO (European Grid of Solar Observations) grid.

References

1. Booch G, Jacobson I., Rumbaugh J.: The Unified Modeling Language User Guide. Addison-Wesley, 1998.
2. Bentley R.D.: EGSO – The European Grid of Solar Observations. In Proc. European Solar Physics Meeting, ESA Publication SP-506, 2002.
3. Cerami E.: Web Services Essentials. O'Rielly and Associates, 2002.
4. Clark M. et Al.: Web Services Business Strategies and Architectures. Expert Press, 2002.
5. Fisher L. (Ed.): Workflow Handbook. Workflow Management Coalition and Future Strategy Inc., 2002.
6. Frankel D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley and Sons, 2003.
7. Georgakopoulos D., Hornick M.F., Sheth A.P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases, Kluwer Academic, Vol. 3 No. 2, 1995.
8. Gibb B., Damodaran S.: ebXML: Concepts and Application. John Wiley and Sons, 2002.
9. HP: HP Service Composer User Guide. Hewlett-Packard Company, 2002.
10. ISO/IEC, ITU-T: Open Distributed Processing – Reference Model – Part 2: Foundations. ISO/IEC 10746-2. ITU-T Recommendation X.902.
11. Klueber R., Kaltenmorgen N.: eServices to integrate eBusiness with ERP systems – The case of HiServ's Business Port. In Proc. Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing (CAISE-ISDO), 2000.
12. Kuno H.: Surveying the E-Services Technical Landscape. In Proc. Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS), IEEE, 2000.
13. Linketscher N., Child M.: Trust Issues and User Reactions to E-Services and E-Marketplaces: A Customer Survey. In Proc. DEXA Workshop on e-Negotiation, 2001.
14. Marton A., Piccinelli G., Turfin C.: Service Provision and Composition in Virtual Business Communities. In Proc. IEEE-IRDS Workshop on Electronic Commerce, Lausanne, Switzerland, 1999.
15. Magee J., Kramer J.: Concurrency: State Models and Java Programs. John Wiley and Sons, 1999.
16. OMG: UML Profile for Enterprise Distributed Object Computing Specification. OMG Final Adopted Specification ptc/02-02-05, 2002.
17. Piccinelli G., Zirpins C., Lamersdorf W.: The FRESCO Framework: An Overview. In Proc. Symposium on Applications and the Internet (SAINT), IEEE-IPSJ, 2003.
18. Piccinelli G., Mokrushin L.: Dynamic e-Service Composition in DySCo. In Proc. Workshop on Distributed Dynamic Multiservice Architecture, IEEE ICDCS-21, Phoenix, Arizona, USA, 2001.

19. Plotkin G.D.: A structural approach to operational semantics. Technical Report DAIMI-FN-19, Department of Computer Science, University of Aarhus, 1981.
20. Sillitti A., Vernazza T., Succi G.: Service Oriented Programming: a New Paradigm of Software Reuse. In Proc. of the 7th Int. Conference on Software Reuse, LNCS, 2002.