DE Services in Ecosystem Oriented Architectures

Paul Malone
Waterford Institute of Technology,
Telecommunications Software & Systems Group (TSSG)
http://www.tssg.org
pmalone@tssg.org

Introduction

This paper introduces the concept of Digital Ecosystem Services deployed in an Ecosystem Oriented Architecture (EOA).

As described in Chapter 3.1, the EOA concept is based on a peer-to-peer architecture that allows the digital ecosystem to be effectively pervasive and decentralised. There is no single point of failure and the ecosystem itself is owned by participant SMEs and not by a governing body or organisation. The knowledge and services are spread across the supporting nodes through a peer-to-peer self-healing architecture.

Services are described in both the technical and business point of view hence allowing semantic and business types of search and discovery.

This paper will describe the requirements of services deployed in an EOA and how the DBE project [1] have realised these requirements for SME service deployment.

1. Digital Ecosystem Services

A conceptual framework for DBE services emerged from the work done by Soluta.net on DBE architecture requirements. This framework is similarly applicable to DE services in an EOA. This can be seen below in Figure 1.

Every DE service is specified using a set of formal languages that aims at defining the business models as well as the technical interface in a platform implementation way. Thanks to the model driven approach taken, business models can be transformed and mapped into platform specific models without specific user interventions. The family of languages adopted defines the service's DNA that fully specify the service and the ability to evolve and adapt.

Each structural component is decentralised. Such an approach potentially allows the ecosystem to be self-healing and survive technical and network failures.

The technical architecture eases the entire integration and adoption process by providing an infrastructure at both sides of the pipe in a consumer-provider point of view. Services already residing in legacy systems can quickly become DE enabled, thanks to the decoupling approach provided by EOA and the MDA (Model Driven Architecture [2]).

A Service Composer allows for browsing of the collected published services and the creation of a workflow that can be executed and published as an atomic service. The chained nature of such new service will be completely transparent to the consumer application or user.

Given the meta-service and meta-modelling approach followed, there is not a single model or service that cannot be replaced or enhanced. No pre-defined or immutable behaviour is coded in an EOA. [3]
2. The Service Factory – Creating and Maintaining the Services

The starting point for a participant in a digital ecosystem is to model and create a DE Service through a Service Factory. A Service Factory is a set of tools to aid the developer in the creation of DE Services, through the association of those services with appropriate models and deploy those services in the Execution Environment. The Service Factory was realised in DBE through the open source DBEStudio Project [4].

The DBEStudio is an Integrated Development Environment (IDE) for the Digital Business Ecosystem (DBE). It was developed using the Eclipse framework [5] and includes a set of eclipse plug-ins that allow business services to be analysed, and corresponding software services to be defined, developed and deployed. When the DBEStudio is launched it is configured by the user to connect to the core DBE Services via a URL denoting the location of a running Execution Environment.

Descriptions of the core plug-ins are provided below.

2.1. BML Editor

The BML(Business Modelling Language [6] Editor plug-in is a visual modelling tool and provides a UML-like Graphical User Interface (GUI), similar to that of well known UML editors. The tool supports the modelling tasks and stores the created models in the DBE Knowledge Base (KB) deployed in the Servent. The current version of the editor supports both the semantic description of the services offered by an SME and the business model of the particular SME. The former provides to the user the ability to create service models according to semantic service language metamodel and the latter to create business models based on the BML metamodel. Both metamodels are described using OMG's MOF 1.4 [7].
2.2. **BML Data Editor**

The BML data editor takes MDA M1 level BML models (created with the BML Editor) and allows the developer to populate these in order to create M0 level instances of those models. These M0 models then represent the business data associated with actual running instances of service business models.

2.3. **Ontology Analysis Tool**

The Ontology Analysis Tool plug-in provides a visual environment based on a UML-like graphical user interface that enables business analysts to deploy domain specific ontologies in order to describe the business requirements of SMEs in the context of the DBE project. The ontology definitions is based on the Ontology Definition Metamodel (ODM) compatible with the OWL, can be represented using XMI (XML Metadata Interchange [8]) technologies and can be stored either locally (in the local file system) or into the DBE Knowledge Base using the JMI (Java Metadata Interface) standard.

2.4. **Service Exporter**

The Service Exporter plug-in enables a user to export a DBE project and deploy it as a DBE service to a Servent. Using a set of wizards the user can add/edit their deployment information. The tool creates a DBE Archive (DAR) file, which contains a particular structure for deployment within a Servent. This plug-in is also integrated with the Metering Wizard to allow users to add metering information at deployment time.

2.5. **Metering Wizard**

The DBE metering wizard is run as an optional element of the Service Exporter plug-in. This wizard allows for the selection of parameters upon which the filters installed in the servent can extract usage data. The SME deploying the service can select methods and parameters of those methods that require metering. This usage data can then be used by OSS (Operation Support Systems) type services installed in the Execution Environment. In DBE, open source accounting services have been implemented and deployed and make use of this usage data in applying charges for services usage as well as providing billing information.

2.6. **Manual Composer Tool – BPEL Editor**

The DBE Composer Tool is a BPEL (Business Process Execution Language [9]) editor to allow for the creation of composed services for execution in the DBE ExE. The design of the BPEL Manual Composer tool centres on a graphical editor and a composition wizard for this composition language. This editor is the core component as it allows the user to graphically design the composed service as a workflow process, while the wizard uses simple rules to help a user to select services and create model structures. The implementation of both the editor and the wizard fully support the BPEL meta-model. The design of the editor provides a 3-view editor where each view has a more abstracted representation of the BPEL model. The intention is to provide two levels of graphical abstraction and granularity to suit both a semi-technical user and a BPEL developer, where the wizard and the graphical editor attempt to address the needs of both user types respectively.
2.7. SDL Editor

The SDL (Service Description Language [10]) editor allows SMEs to define their services from a technical point of view. The editor provides a graphical means of defining service interfaces and expresses those interfaces via XML instances of the SDL schema.

2.8. SDL2Java Compiler

This plug-in takes the XML SDL instance created by the SDL editor and generates a set of Java packages and interfaces which need to be implemented by the developer in order for the service to be deployed successfully in the SBE Execution Environment.

2.9. Service Manifest Composer

The Service Manifest composer is responsible for the creation of the Service Manifest (SM) [11]. The Service Manifest acts as an advertisement for a deployed service and contains both business and technical models of the service instance. The Service Manifest Composer is responsible for the creation of this from the models created from the other plugins. This SM is deployed with the service and inserted into the Semantic Registry (Service Registry). See the next section for details.

3. Execution Environment – Deploying the Services

As described in the chapter on EOA, an execution environment for DEs must provide certain functionalities; a suitable peer-to-peer network, a service container, a service composition engine, a model repository and a service registry.

![Figure 2: Distributed Infrastructure of EOA](image)
The distributed infrastructure of the EOA can be seen in Figure 2. The nodes are connected via the peer-to-peer overlay network. Each node contains the core functional components required for the EOA to be successful. Each node also hosts services offered by SMEs. The services have the ability to migrate, providing high availability of service provision in the event of a node failure or downtime.

Each of the supporting core components is briefly discussed below and the DBE implementations also referenced. Detailed descriptions of these implementations are available in the chapter on Distributed Infrastructural Services.

3.1. Service Composer

The composer provides the ability to orchestrate and manage the execution of composed service chains. The DBE implementation integrated the open-source workflow engine ActiveBPEL [12] and extended it with a custom invoke handler to deal with invocations. This provides the advantages of a standards-based workflow description and execution with the added bonus of DBE peer-to-peer interaction.

3.2. Model Repository

The model repository is a business model container. Services deployed in the ecosystem are associated with one or more of these models. The preferred encoding option of models in the registry is XMI (XML Metadata Interchange)[8] (see chapter on Ecosystem Oriented Architectures). The DBE implementation, developed by the Technical University of Crete, is called the Knowledge Base and provides distributed persistence satisfying the OMG MDA [2] approach taken in DBE.

3.3. Service Registry

The service registry is a repository for references to deployed services in a digital ecosystem. Each entry is associated with one deployed service and contains information on business models, technical specifications, business data and the service end point. The DBE implementation of this component, developed by the Technical University of Crete, is the Semantic Registry, which is used to store a Service Manifest [11] per deployed service.

3.4. Peer-to-Peer Overlay Network

A peer-to-peer overlay network is essential in providing a digital ecosystem with the assurance of no single point of failure and robust distributed knowledge and service provision through a distributed set of collaborative nodes. This approach improves routing resilience to node failure. A suitable implementation needs to be self-healing and autonomically adaptable to the changing nature of the requests and the traffic.

FADA (Federated Autonomous Directory Architecture) [13] was the initial peer-to-peer implementation deployed in DBE. FADA emerged from the European project Fetish. Trinity College Dublin also developed a DHT (Distributed Hash Table) peer-to-peer implementation based on their peer-to-peer architecture design for DBE [14]. More details of both these implementations are available in the chapter on Distributed Infrastructural Services.
3.5. Identity

A fully distributed identity management system is essential for providing trust among the participants in digital ecosystems. Identity constitutes one of the basic building blocks for providing accountability functionality to B2B transactions. Services need to be associated with an identity of the service provider and service consumers need also to be identified for accounting and access control purposes. However, creating a decentralised, robust and trustworthy identity management system with no dependencies on third party certificate authorities is a challenging proposition.

In the DBE project, Trinity College Dublin developed a core identity component overlayed on top of the DHT implementation. This constitutes a decentralised solution that provides the redundancy and management features inherent in the DHT. The system has the ability to verify keys associated with service invocations.

4. Conclusion

Successful digital ecosystem service deployment in EOAs requires a set of mechanisms for the definition of business and technical models, the creation of service interfaces based on these models and robust decentralised service hosting. These mechanisms have been described in this paper, together with descriptions of how these mechanisms have been realised in the DBE project.

5. Acknowledgements


References


