D.B.E.

Digital Business Ecosystem

Contract n° 507953

WP 21: DBE Architecture Requirements

Del 21.2: Architecture Scope Document

Project funded by the European Community under the "Information Society Technology" Programme.

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: D21.2 Release C

Due date: May 2005

Delivery Date: 6th of June

Short Description:

Architecture scoping document

Partners owning: Soluta.net

Partners contributed: Sun, ICL, LSE

Made available to: All DBE partners and the EC

Version	Date	Author	Description
01.00.00	December 2004	P.Ferronato	Started
01.50.00	January 2005	P.Ferronato	Released to internal reviewers
01.50.00 A	February	P.Ferronato	Release A
01.50.00 B	April	P.Ferronato, A.Neagu	Release B, added MDA section
	May	P.Ferronato	Release C, added use cases for SF and ExE, Executive summary, and EvE description

Quality check

1st Internal Reviewer: Paul Krause (Unis)

2nd Internal Reviewer: Paul Malone (WIT)

Table of Contents

Figure Index

1 Changes

Version	Date	Description
01.00.00	December 2004	Started
01.50.00	January 2005	Released to internal reviewers
01.50.00 A	February	Release A
01.50.00 B	April	Release B
	May	Release C

"[The architecture] is a long and sometimes painful succession of sub-optimal decisions made partially in the dark"

Grady Booch

2 Executive Summary

The Model Driven Architecture defined by OMG has been chosen as the main reference approach for driving the implementation process and strategy in the DBE project. It provides guidance for the definition of service models specification and transformation, it is defined by a standards body which has defined several important standards including CORBA and UML. The entire architecture has been effected by this decision and this choice is evident in the terminology as week as the implementation.

The DBE architecture is made of three distinct environments: Service Factory (SF), the Execution Environment (ExE) and the Evolutionary Environment (EvE).

The SF is represented by a set of tools that allow SMEs to describe and publish their services. The SF leverages the use of models editors, regular Java based development environment and a repository where to publish and retrieve service models and data.

The ExE is a sort of distributed virtual machine or Networked set of Operative Systems that allow services to be published, discovered and consumed over the Internet. In addition the ExE provides a set of structural features like authentication, authorisation, privacy, transactions, logging and so forth.

The EvE is an underlying environment where complex services are transparently created and chained from the basic one found in the ExE. Such services are created for supporting un-fulfilled business requirements. For example the EvE could realize that in a specific region there is a recurring requests for deratization. The information for achieving these goals comes from the usage data provided by both the SF and the ExE. The services which are created are in the form of a specification rather then an implementation (which would be rather impossible to achieve at this state of the art in artificial intelligence) and will allow SMEs to implement it and successively other SMEs to consume it.

Services in the DBE are essentially described in the facets: business and computing. The first aims at describeing the service in purely business terms like the goal of the firm, its products, its offers, the price models, the business model and strategy. All this information is defined in terms and concepts that are abstracted from any implementation specification. The goal is to allow not IT specialists to define and understand such a specifications and at the same time to make it precise and complete in order to be computable. The computable form of the business models can provide the ability to create automatic contract negotiations and agreement creation; another goal is to compute the merit of business offers.

The computing models associated with the business specification allows a businesses' offers to be consumed via on-line services. This model is expressed in a way that allows non predefined IT functional models to be integrated in the DBE. The goal is to be as little invasive as possible and hence not to enforce compliance to DBE functional and technical models. Once again the MDA approach is of help here because we ease the transformation of the computing model.

The business models and the computing model together with instance data like location, name, costs makes up the "service manifest" that is hence the self contained descriptor of the service.

3 Introduction

The DBE project is a natural extension of the Fetish project. Even though there is a continuous line that connects the two projects, the DBE has not much in common: it is a very demanding new project with very challenging objectives. It is important to note that there is (in contract to the Fetish project) an important trilogy: science, business group and IT. This project and the architecture, as a consequence, have to manage and handle these three different, not always cohesive visions. The architecture is asked to provide the guidelines to build such a complex artefact.

This document

This is a living document, and will be maintained and enriched until the 18th month.

There will be three major releases:

- 1st release A, 7th of January – Reviewed by 17th of January
- 2nd release B, 18th of February – Reviewed by 28th of February
- 3rd release C, 22nd of April - Reviewed by 2nd of May

3.1 DBE as a service distribution bus

The major role of the DBE is, in simple terms, to support service discovery and execution for SMEs (in relation to the Information Level Exchange [HS2002], the DBE is located at level III) with a unique approach that focuses on the evolutionary/ecosystem approach. The DBE hence assumes that services already exist and are either provided by a human being or supported by an information system in an environment that it is not under control of the DBE; it is out side of its range.

In biological terms we can say that the DBE is not involved in the creation of individuals but instead is supporting and helping their interactions.

The various software artefacts that the project will deliver are conceived and realised with the goal of helping evolution and integration but not of "developing services". In this sense we have to clarify that the DBE environment devoted to service creation is taking care of describing it and generating all the structural code needed to manage it, but not to implement how the request is going to be fulfilled; this is delegated to the SME back office.

The most correct definition still resides in its name: digital business ecosystem.

3.2 Naming convention

In this document and in the subsequent architectural documentation that will be delivered, the term "BML Editor" will be replaced by "Integrated CIM Editor" (the same applies to "SDL Editor" that becomes "Integrated PIM Editor"). We believe this definition clarifies some misunderstandings that have recently emerged between a specific modelling language and the reality that is going to be described with that language. A sufficiently abstract language can be used to model all the various facets of an SME, mostly ranging from business to computing. In this sense, for example, UML can be used to model the internal system specification of an information system as well as its functional specification. Being so generic is a good thing for UML, but the business modelling

language to be defined for the DBE has to be more more specific and closer to the business.

As a matter of fact it is very hard, and probably not even helpful, to have a language that is limited to only one domain, it can always be used for other purposes unforeseen at the definition time of the language.

The BML, as conceived in the DBE, has the goal of allowing the description of SMEs, their services and products. This happens currently in pure business contexts during, for example, business meetings with business people that have the onus of evaluating the feasibility of starting an agreement or a partnership. These meetings do not mention anything related to computing or information science (this applies also, at the proper business level, to software developer companies). Since we are developing a computing infrastructure for SMEs that might not even have an IT system, we must strictly focus on a modelling phase that does not require computing-based expressions.

Given the above considerations, we believe that we have to replace the term "BML Editor" with "Integrated CIM Editor". In this way, any possible confusion between the underlying languages used and the goal of the modelling phase is clarified. Hereby it is important to highlight the fact that the first, most important phase of the modelling process needs to be Computationally Independent[1].

1 At any rate, it is not mandatory to disclose this name to the end-user of the DBE; for usability reasons it could have a different name in the final delivered software tool, depending on the result of the interaction design.

4 Architectural Principles

"Believing in architectures, is like believing in God. It makes you feel optimistic."

[unknown]

The nature of the project requires the use of several levels of abstraction for the service specification, layering, run-time configuration, self-description and reuse of open standards. The core architecture that is described here will be a base for the architecture of all the sub-projects of the DBE. The term sub-project is reductive for the DBE parts since they are all very sophisticated and are full-featured projects by themselves: the term sub-project has to be considered in a relative sense .

4.1 Pareto's Principle

We assume (reformulation the Pareto principle) that 80% of a project is trivial and that 20% is complex, also that 80% of consequences result from 20% of causes. For this reason the 80% of the effort should be concentrated in that 20% that matters. For this reason the architecture has to strongly support that 80% with default basic/simple/automatic mechanism and to enable the other 20%.

It would be silly to jeopardise the architecture by pushing a high level of sophistication to address the easy 80%. In this way we aim at having a fast/reliable infrastructure for the majority of the services and at the same time to support sophisticated services by allowing customisation.

4.2 Architectural Viewpoints

Often the Architecture in IT is referred to as an activity that manages risks and delays. But this effort is not yet as well established as other IT disciplines like requirements gathering, analysis, design and the software life cycle. An architect has to supervise technology adoptions, software distribution, network topology, process development, tool selections and so forth. The are a lot of areas which are subject to investigation and there is the need to separate the concerns and efforts.

It has been recognized that we can identify:

- structural architecture: how to make good use of the technical architecture
- technical architecture: which technologies to adopt
- functional architecture: what the system has to accomplish
- management architecture: how to organize the work. We are not going to address this last point.

Structural

The structural architecture is the architectural viewpoint concerned with the set of architectural decisions, patterns, guidelines, and standards required to build a component-based system that conforms to non-functional requirements. It addresses the set of things needed to build a scalable and efficiently performing system.

This includes the set of available generic services and is centred on structural components; for example, error specifications and error handling, business data types, generic structural patterns for distributed systems and for the persistence architecture. Other application architecture aspects are designed to take into account the nature of service components and the need to preserve autonomy of development and system-level development integrity.

Technical

The technical architecture is the architectural viewpoint concerned with the set of tools, the user interface framework, and any other technical services and technical facilities required to develop and run a component-based software system.

This viewpoint generally includes the Integrated Development Environment (it is named the DBE Studio), which is the term that is used for the set of tools, compilers, editors, models and so forth required to develop a component-based system. The implementation of the technical architecture viewpoint is mostly related to the environment where services are deployed and executed (the Service Execution Environment).

Functional

The functional architecture is the architectural viewpoint concerned with the functional aspects of the system; that is, with the declarative and behavioural specification of a system that satisfies the functional requirements. In the DBE, we are concerned with the functional specification of the DBE itself and with the SME Business services. What distinguishes the two aspects is that the latter requires to deal with meta-functionalities

and hence to define meta-service specifications. One of the most challenging aspects of the project is to abstract the SME needs in term of service and business specifications in order to derive the actual functional specifications of the tools that will support the entire life-cycle of a digital business ecosystem.

The DBE, as pointed out in the "DBE Architectural Requirement" document, is not playing at level II in the Information Level Exchange [HS2002], where the functional specifications are there to be discovered and where the development tools are there to be used. In these kinds of projects the complexity is mostly due to the horizontal viewpoints, the extension of the specifications, the quantity of processes, messages and data. Any metadata aspect is primarily related to relational databases where tables describe tables and this is achieved in almost an automatic way. The functionalities of ERP systems seem mostly fixed if compared to the dynamic nature of an ecosystem with thousands of individual applications aggregated in ever-moving and adapting populations. Even if we may reply that ERP systems are characterised by having a non-static specification of business processes, we soon discover that these processes are a rather static set of process names; even if the implementation workflow is dynamic, the kind of processes are not. As a consequence, modelling is the main effort of the project team in these cases while in the DBE, or in general in ILE III (Ref. DBE Architectural Requirement) kinds of projects, the main effort is metamodelling and modelling is left to users. Referring to the MDA abstraction layers, the DBE architecture is directly involved in creating a support infrastructure for creating M2 instances from M3, with the goal of helping SMEs to create M1 model objects from M2 by themselves and finally to let final users to create M0 from M1.

From the perspective of this document and of the Architecture in general, "functional specification" is related to the DBE supporting infrastructure.

4.3 The component based approach

The component based approach has been first introduced by Oliver Sims in his well-known book "business objects" [SIMS94]. This way of writing software has the goal of realizing autonomous software artefacts (reusable and run-time pluggable) which reflect the business element they represent.

Components

The main definition in the component-based development approach is the **Distributed Component**[HS2002]. It is a design pattern that describes how to realise an autonomous software artefact that can be deployed as a pluggable unit into a run-time execution environment. This pattern is aimed at supporting high-productivity development for large-scale distributed systems. It is implemented by adopting a distribution implementation technology like RMI, CORBA, .NET, WS and others.

Generally when a distributed component is deployed it immediately becomes network addressable. However, the distributed component term is normally used for both the design pattern and the implementation, unless there are reasons for making a stronger distinction. The DBE execution environment is the reference middleware platform that supports the deployment and the execution of all the structural and functional services [FER04] implemented as distributed components.

In the DBE, the distribution technology adopted is called the Nervous System and is implemented with FADA. Usually the Distributed Component is conceived to work in an Intranet and it needs to take into account the troubles and hurdles of the Internet when used in an Extranet.

Each distributed component is developed using Java object-oriented **language classes**. Language classes are used by a developer to build a distributed component. A language class is not considered a component, but it is certainly a software artefact of fundamental importance.

A **business component** is the software implementation of an autonomous concept or process. It consists of all the software artefacts necessary to represent, implement, and deploy a given business concept as an autonomous, reusable element of a larger distributed information system.

A **business component system** is a set of cooperating business components assembled together to deliver a specific solution to a functional problem. A **system-level component** is a business component system that has been designed to interoperate with other systems; it is constructed as a component in its own right. For this reason a Business Component System, when assembled, deployed, and network addressable, is equivalent to a System-Level Component.

A **federation of system-level components** is a set of collaborating system-level components connected to address the information processing required by multiple external (to the federation) components perhaps belonging to different organisations. In the context of the DBE there is the perspective that in the long term there might be other external communities of services (pertaining to different organisations) that need to cooperate and exchange messages. In order to achieve this scenario the DBE and the other communities shall provide a gateway service to bridge the federation of system level components. This kind of component is not addressed for the time being by the DBE consortium.

> **Note**: the **proxy** is a term that has a strong technical flavour since it is often related to a specific middleware like Jini, .Net or FADA itself. Both the proxy and the distributed components terms are a platform independent concept. At any rate, a proxy is conceived to be distributed when a consumer component asks to invoke it, while a distributed component is statically distributed at deploy time.

Distribution Tier

A business component (and, applying the definitions of the above chapter, an entire information system) is made of mainly four tiers: user, workspace, enterprise, and resource tiers[HS].

The **user tier** renders the business component on a screen and is responsible for the communication with the user. It is the responsibility of the user tier to implement and execute the panels and the user interface controls for a particular business component. This tier is placed as close as possible to the underlying GUI system. The user tier can also provide programmatic APIs and not only be available as a GUI subsystem. The DBE Studio for example makes intense use of plug-in APIs in the GUI.

The **workspace tier** supports the user tier as a local working area. It implements local self-contained business logics and is responsible for communicating with the remote enterprise tier to access any enterprise-level resources and intelligence needed to support the user's working session. This well-known separation of concerns, i.e. keeping the business logic local to a single user separated from the business logic related to the enterprise, allows for different kinds of optimisation, and it supports the highest levels of performance and scalability. An access to the enterprise resources (intelligence, policies, rules and data) should be the sole responsibility of the workspace tier that can coordinate and optimise the low performance of the remote channel (i.e. the Internet for the DBE). If the user tier needs to render data coming from the enterprise, or if data entered by the user must be used to update the remote database, then the user tier must always make a call to the workspace tier.

The **enterprise tier** contains the core aspects of a business component. It implements the enterprise-level business rules, complex pre-validations, and interactions between other enterprise components. This is the most important tier because it contains most of a business component's logic and because it is responsible for addressing strategic requirements, performance, scalability, integrity, reliability, access control, privacy and the information system. The enterprise tier has generally to be accessed concurrently by users, and it requires persistence, state management, security in general, and transactional services. The technical architecture of the DBE supports the enterprise tier by providing these support services. Such services are meant to be used both by the structural services of the DBE and by the generated business services.: the DBE intends to provide to SME users and developers the same support services that are used internally.

The **resource tier** manages the physical access to shared resources. Data in shared databases is the most common form of resource, and this tier is responsible for bridging and mapping the functional developer's model inside the enterprise tier with the actual reality of the database implementation. Furthermore, this tier is also responsible for shielding the functional developer from the peculiarities of the various database management systems. For example, the coding of embedded SQL to access relational data would be in the resource tier.

These tiers are logical tiers that can often converge to a physical tier as a network address: on the other hand a tier can be further split into two separate physical tiers. More often the user and the workspace tiers are combined in one single physical tier.

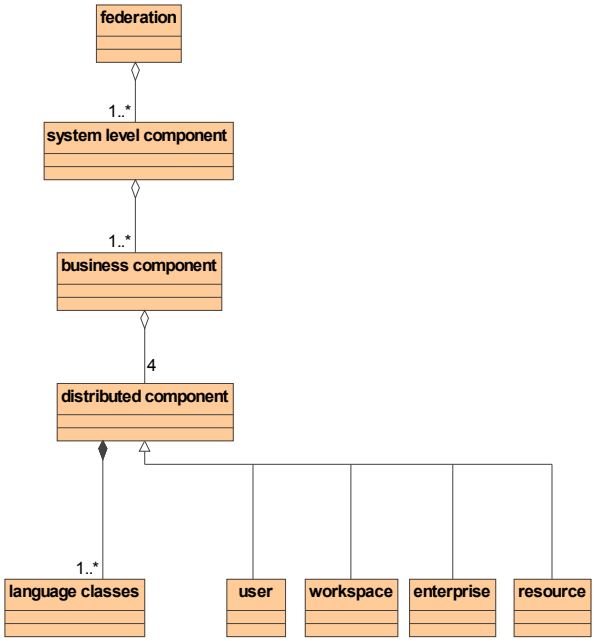


Figure 1 - Component Framework

4.4 MDA Approach

The MDA approach to system specification focuses on the separation of concerns by separating the specifications of functionality from the specifications of the implementation of that functionality, for a specific platform. It also defines the means to transform independent system specifications to platform specific implementations. In this way, the model traceability in the software development process[2] is maximized.

MDA

MDA is an architectural approach in which modelling languages are used as programming languages. This can greatly improve the productivity and the quality of a project. It is model driven in that it uses models to describe and control all the activities involved in defining, building and maintaining a system.

MDA uses the MOF's (Meta Object Facility) concept of structuring the information in levels of abstraction. At each level of abstraction the information is seen from a viewpoint and shaped into models accordingly. Three kinds of viewpoints are used, which leads to three types of models:

- a computational independent model (CIM)
- a platform independent model (PIM)
- a platform specific model (PSM)

The transformation of a PIM to a PSM is formalised as a mapping. If this transformation can be automated, a huge amount of time is saved, time used mostly to rewrite a system implementation, when its technical specifications have changed.

To use MDA to define a system means first to create a CIM in a language familiar to the business representatives, and from this to create a PIM which contains the functionalities of the system without the details of its implementation. The next step is to choose the

2 In MDA terms this is called Model Driven Development.

platform on which the system will function, and to define mappings for transformations that will create the PSM from PIM for the chosen platform. In this way, if something changes in the business requirements of the system, it is sufficient to update the PIM, and using the defined transformations the PSM is recreated.

It is not our goal to describe the MDA in detail. We describe the principles and features that are relevant to understand the DBE features and architectural choices.

MOF

The Meta Object Facility (MOF) is one of the key foundations of the MDA. It is a lightweight modelling language intended mostly to define other modelling languages which are dedicated to a specific domain (UML for class modelling, and CWM for databases for example). It can be considered a hypothetical "Universal Design Language" (UDL).

Levels of abstraction

In order to structure the information, MOF defines four levels of abstraction. The Figure Figure 1 - Component Frameworkabove is a view of those levels used to define the modelling languages for DBE.

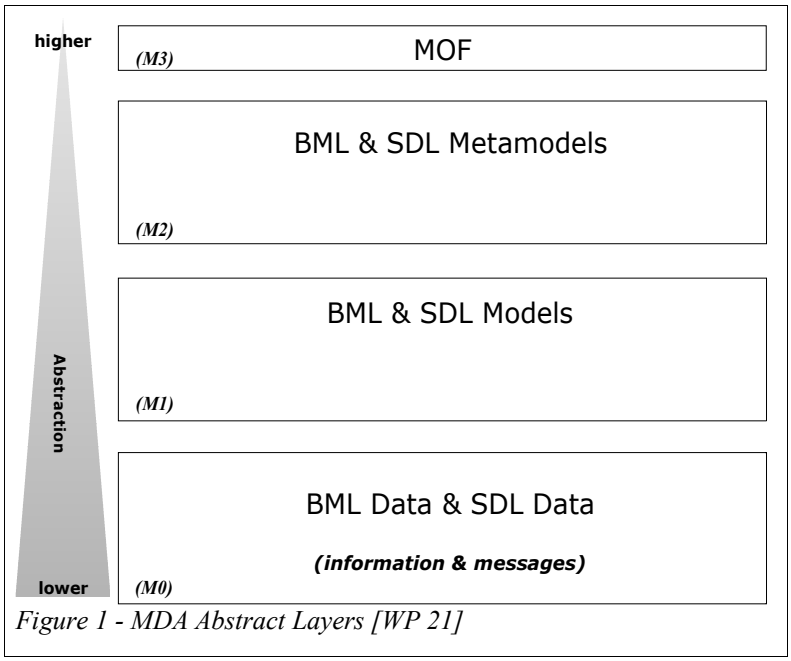


Figure 1 - MDA Abstract Layers [WP 21]

"Level M3 is MOF" [MDA]. The M3 meta-metamodel level is the most abstract level, MOF is defined as self describing.

The M2 metamodel level consist in instances of MOF constructs. Here are defined the modelling languages which will be used for modelling: e.g. Use Case, Actor, Class, Interface and so forth.

The M1 is the model level that consists of instances of the M2 level constructs. It consist of models which describe what an application deals with at a certain level of abstraction: e.g. Supplier, Seller, OrderManager, Customer and so forth.

The M0 is the data level, it contains the information defined by the M1 level models: e.g. Customer=ID96762529, Seller="ToolsRUs", priceModel="HighSeason" and so forth.

Viewpoints

A viewpoint on a system is a technique for abstraction, using a selected set of architectural concepts and structuring rules. An instance of a viewpoint is a view, which is the representation of the system from a specific "point of view". As defined above, MDA is currently using three types of viewpoints for a system representation which leads to three types of models.

CIM Viewpoint

The Computational Independent Model or CIM, deals with the business requirements of a system without showing details of the solution space: what has to be solved and addressed instead of how the requirements will be delivered. A CIM can be viewed as a domain model and its specifications should be written using a vocabulary specific to the domain in question. It intends to be a bridge between the experts of the domain, its requirements and the designers.

PIM Viewpoint

The Platform Independent Model or PIM describes the functionalities of a system without showing the details necessary for the specific implementation platform. The description does not contain the part that changes from one platform to another. It is used to realize the functionalities in a computational form described by the CIM but in a implementation neutral form.

To be platform independent a system can be usually targeted to a technology-neutral virtual machine, which is defined independently of any specific platform (Java is a good example of that). But the platform independence is a relative concept, for example there are other kind of virtual machines that stay at the same abstraction level of Java (C# for example). CORBA for example was considered to be platform independent, because it was able to deal with different kinds of legacy systems. Nowadays, together with Web Services, it is considered a kind of implementation technology. In the same way in the late 70s Fortran compilers were considered platform independent (although the term used to describe such features was different) for its ability to generate code for different families of CPUs.

As a matter of fact, given the relative nature of the PIM concept, the DBE itself can be considered an implementation platform, but since it is the top most abstraction level in our context, it is considered as platform independent.

PSM Viewpoint

The Platform Specific Model or PSM combines a PIM with additional details on the use of a specific platform by the system. It combines the specifications from the PIM with the information that describes how the system will use a particular platform. Then, a PSM is realised from a PIM trough a transformation, which is an aggregation between the PIM and information needed to implement the system on a specific platform. One of the goals of MDA is to automate this process with the use of specific tools.

In DBE terms the PSM is a mix of technologies that comprise Java, P2P, XML-RPC, XUL, JMI and others.

Model transformation

As we have described above there are several model transformations in MDA. For example PIM to PSM, but there can be also transformations in the same abstraction layer. In more detail, model transformation is the conversion of a model of a system to another model of the same system. This can be done through a variety of methods which we briefly describe:

- marking. The PIM is annotated with platform specific marks, and from this the PSM is generated.
- metamodel transformation. The PIM is created using a platform independent language specified by a metamodel. A platform for which a metamodel exists is chosen. Then a specification of the transformation terms of a mapping between metamodels, is used to create the PSM from the PIM. This is the most used approach to model transformation in the DBE.
- model transformation. The PIM is created using platform independent types specified in a model (the elements of the PIM are subtypes of the platform independent types). Then a specification of a transformation for the chosen platform, specification defined in terms of a mapping between the platform independent types and platform dependent types, is used to create the PSM from the PIM.
- pattern application. In addition to platform independent types, patterns can be used to define the PIM. The PSM is created from the PIM using a transformation specification defined in terms of mappings between the platform independent types and platform dependent types, and between platform independent patterns and platform specific patterns.[3]

A model transformation in MDA can be done manually, automatically or a mixture of both.

- Manual transformation: this is not different from the current process of software design. What is new is the recording of transformations and the explicit distinction between PIM and PSM.
- Automatic transformations: there are situations in which a PIM can contain all the information necessary for the system implementation, and there is no need to add extra data in order to be able to generate the code. If this is not achieved, there are intermediate degrees of automation that can be used. Those are the UML profiles, and patterns and markings.

UML profiles

UML is a flexible modelling language and provides build-in mechanisms for extension (stereotypes and tagged values). These extension mechanisms allow the modeller to define and use additional modelling constructs besides those defined by UML. An UML profile is a set of extensions of the UML metamodel.

A PIM prepared using a platform independent UML profile can be transformed into a PSM defined with the use of a second UML profile, this time platform specific. This transformation can imply the marking of the PIM with marks from the platform specific profile.

3 There is also “model merging” which is not listed here because it is not relevant in this context.

Patterns and Markings

As stated before, patterns and markings can be used to specify a mapping for a model transformation.

Reasons for using the MDA approach for DBE

Since the DBE is an environment where the essence of business and services are described through models and model transformation, the choice of MDA was rather a natural decision.

The MDA helps to address the gap between formal modelling and practical software applications. It gives the possibility to automate the generation of PSMs from PIMs. By the use of MOF and its levels of abstraction, the MDA offers the possibility to work simultaneously with many domain dedicated modelling languages, and the ability to define new modelling languages when they are needed.

An MDA compliant approach will also allow for interoperability with existing SME models and standard bodies; for example the DBE is already able to "import" and transform the Open Travel Alliance specifications and WSDL.

MDA for DBE implies an emphasis on the automation of the transformations of PIMs to PSMs. As an example, the SDL editor is generated automatically from its metamodel (using a mapping). Whenever the metamodel needs to be updated, the SDL editor is promptly re-generated.

The MDA serves also as a common ground for project participants since it provides a vocabulary, a reference standard for model encoding and processing, a standard interface for accessing MOF based information and to be able to access a wide network of communities sharing the same vision and tools.

5Functional Architecture

"There are no facts, only interpretations."

[Friedrich Nietzsche]

A global separation of concerns in the DBE can be identified by:

- Service Factory (SF);
- Service Execution Environment (ExE);
- Evolutionary Environment (EvE).

In Figure 20 below we have on top the layer representing the real business network that is made of persons and tangible goods, the middle layer represents the two environments that are visible to the SME users, Service Factory Environment (SFE) and the Service Execution Environment (ExE). Here it is evident that the data flow and messages are direct from the SFE to the ExE. The lower layer represent the Evolutionary Environment that has to implement the automatic evolutionary feature of the project; this environment is transparent and not visible to the user community.

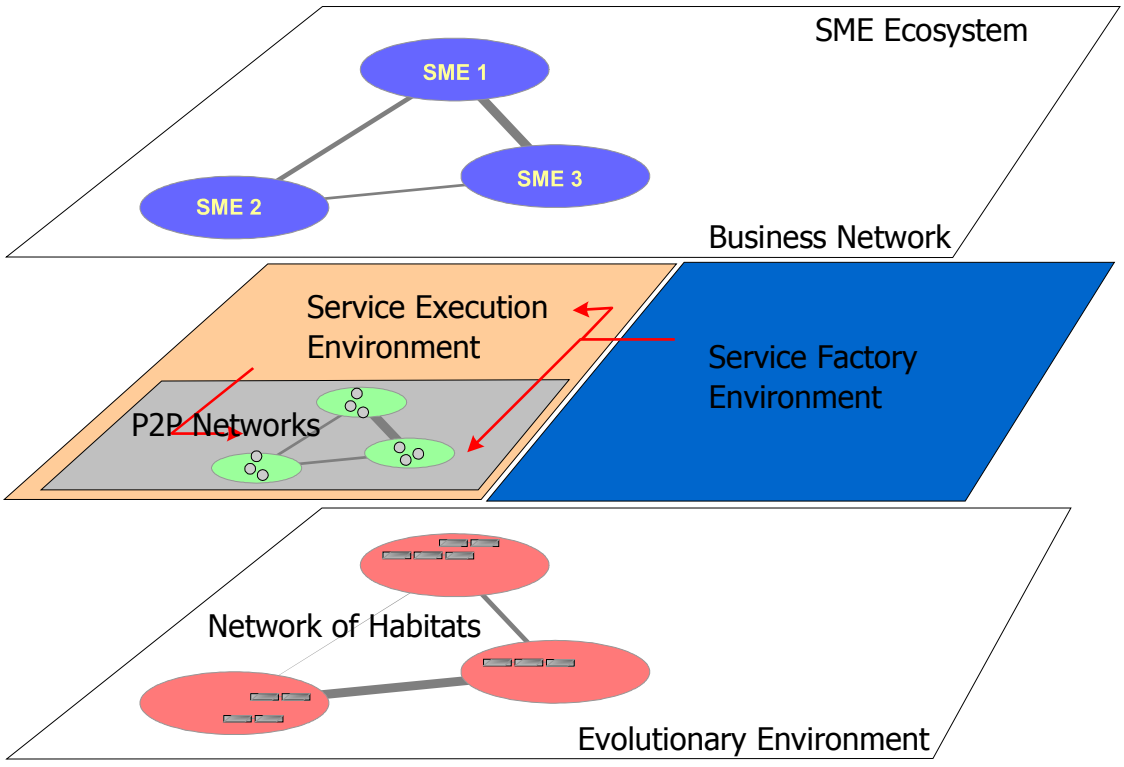


Figure 2 - DBE Environments (adapted from Salzburg TechnicalUniversity)

The Service Factory Environment is devoted to service modelling and definition[4]. Users of the DBE will utilise this environment to describe themselves, their services and to generate software artefacts for subsequent implementation, integration and use in the ecosystem. This phase requires to make use of abstraction, reuse strategies, business and technical modelling. While in regular projects this phase is realised by third-party tools, in the DBE we have to build our own tools. Moreover, this process is usually one-shot; it terminates

4 The SFE is sometimes referred to as the "design-time of the DBE", which is an acceptable definition. Sometimes it has also be called the "development environment", but this is a reductive and inaccurate description.

with the deployment of the application and this represents the end of the project, even though some significant effort remains for maintenance purposes. Here, on the other hand, the "Service Factory" is supposed to be never-ending, it is part of the run-time of the project: this concept is shown in Figure 4 below. The DBE as an instance, i.e. the "DBE Community Instance", is composed of the two sequential environments (the EvE is not shown).

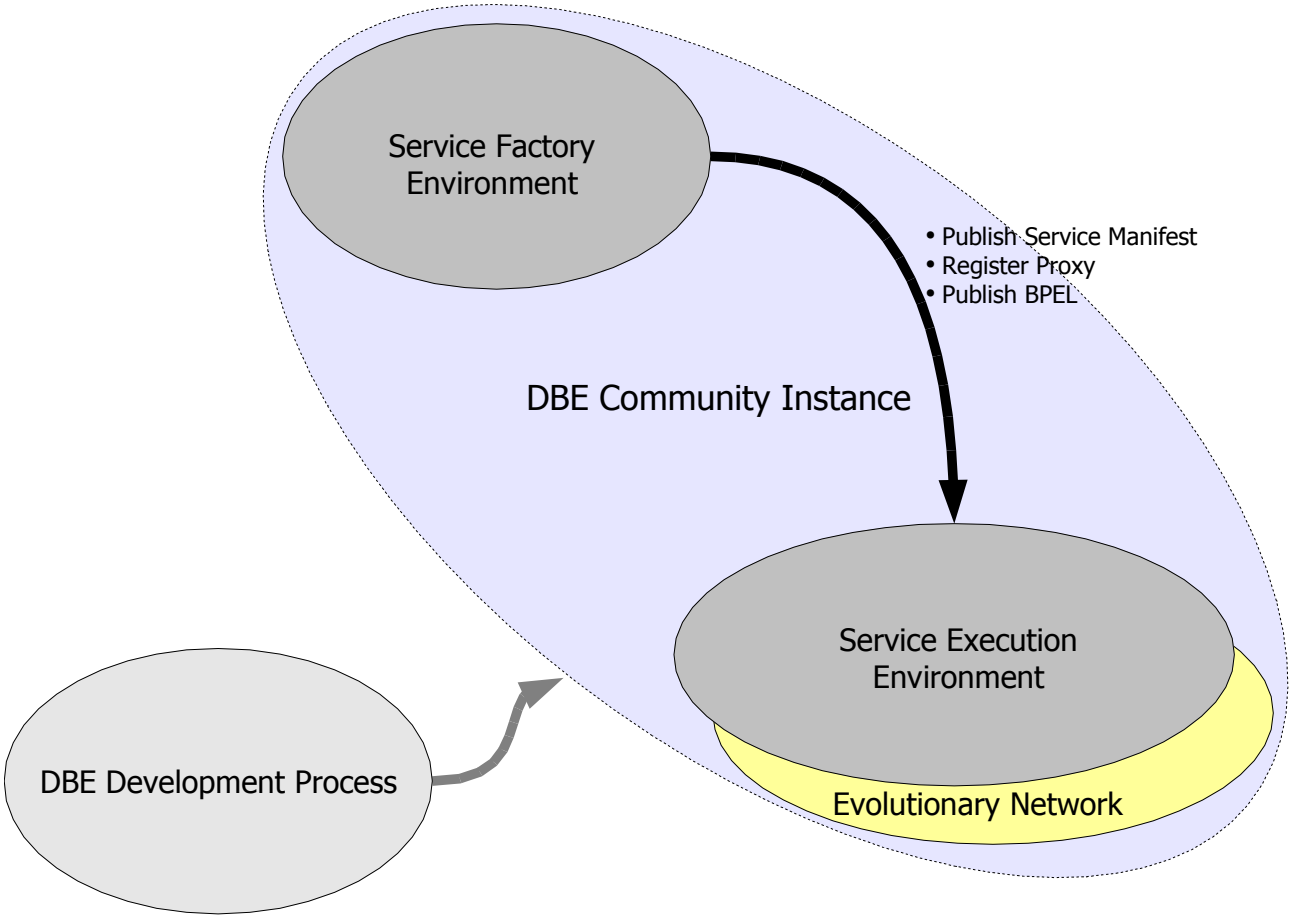


Figure 3 - The three DBE environments

The service execution environment is the ecosystem itself where services are advertised, negotiated, get connected and are executed.

The Evolutionary Environment is the realisation of the Science work on distributed evolutionary computing. It will provide a scalable solution to the problem of providing aggregated services to meet user requests. Each SME user has a Habitat on the habitat network, within which their own services are distributed and from which they can find services of other SMEs. The habitat will also perform evolutionary computing upon the services available within the habitat to construct solutions to SME user requests.

The EvE represents an alternative way to distribute, locate, and aggregate services. When a service is constructed from the Service Factory, it will be distributed from the SME's own habitat instance (see Section 4.5) automatically across the Habitat Network, becoming available for evolutionary service composition at other SME user habitats. This is possible as the habitats are all interconnected, dynamically clustering around business sectors and supply-demand relationships. So distributing a service from the SME's own habitat will provide the optimal distribution of the service to the users who are most likely to consume the service, based on past experience.

The Distributed Intelligence System will augment the EvE, enhancing and optimising the evolutionary process by the application of intelligent intervention to accelerate evolution. This will provide a facility similar to the real world when humans breed plants or animals for specific features and qualities, encouraging reproduction between individuals that will lead to the desired features and qualities. The DIS will provide a Clustering Catalyst in each habitat for the determination of clusters in evolving populations. Then crossover (genetic recombination) of individuals within these clusters can be enforced to accelerate evolution.

5.1 Service Factory Environment

The SFE is the place where the SMEs "create offers" that can be related to services or products. The creation process requires the description of the many facets of the offer, which span from the business motivation of the firm to the software interface parameters that enable it. Rather then being just an extended development environment that supports coding and testing with a programming language, the SFE specifically supports the complex and articulated effort of describing the offers from a business perspective that, in the current Service Oriented Architecture (SOA), are only marginally treated. The SFE, in contrast, widens the horizon by taking into account the parts that are strategic in real business transactions, i.e. business location, business motivation, business process, events, processes and so forth. The current effort in IT is mostly aimed at describing services from a computing perspective, sometimes enriched with semantic specifications, and it is only marginally interested in describing it from a pure business viewpoint[5]. The availability of an extended "business" description allows the realisation of intelligent search mechanisms that aim at automating the B2B discovery, negotiation and consumption transactions. Enabling business and semantic discovery with pure Web Service technology is awkward, analogous to evaluating suppliers by inspecting and evaluating the trucks used to deliver the products.

The SFE does not only provide a set of visual tools for modelling and developing, it aims also at giving seamless support in the entire path needed to describe the offer that can be a product or a service (often these two terms converge). This path is made up of mostly sequential phases that require different actors with different objectives and skills. These phases, described below, start from pure business modelling down to platform code development, and the SFE aims at supporting these transitions through semi-automatic model and code generation.

The Service Factory Environment can be divided into three mostly sequential activities:

1. business specification (CIM): where the business model of the service is realised. This is further and logically divided in:
    1. business modelling
    2. service modelling
2. interface specification: where the service is technically profiled and where extra-functional specifications are given (PIM)
3. coding (PSM): where the service (either in the role of consumer or provider) is implemented (Java code, either DBE generated or hand written). EAI, integration.

5 Several initiatives are emerging at the moment both from standard bodies and from the industry like BRBR sponsored by Unisys..

This set of models needs to be semantically consistent and rich in order to be able to extract coherent information: this is provided via an ontology system that is the fourth, horizontal phase. It does not belong in the linear sequence of the three phases described above (Ref. Figure 5 - Dependencies between languages on page 25).

CIM, PIM and PSM are used for supporting reasoning and evolution of services, during the discovery, negotiation or execution, and they are a means to automate the code generation phase: at the time being there is not an effort in the consortium for supporting action languages.

Such phases are supported by the following visual software tools:

- the "Integrated CIM Editor"
- the "Integrated PIM Editor"
- the "Integrated PSM Editor".

The first is the place where business users will create the business models that will be the foundation of the subsequent phase, where the services will be defined from a Platform Independent (PIM) perspective. The CIM environment is not related to IT terms like exception, parameter, string, class or something; it is conceived to be used solely by business analysts and the SME users themselves. This last goal depends on the interaction design of the tool, the metamodel expressiveness, and the availability of "wizards" that can derive the most appropriate models from an "expert system"-based interview.

The second tool helps the definition of the computing functional interface that supports the part of the business that has to be automated. The actor involved here is essentially an IT person that need not have specific skills in the underlying adopted technology (i.e. the implementation Platform).

> **Note:** the PIM environment is itself, on the other hand, DBE-specific (relatively speaking we might say that it is a PSM itself).

The PIM editor makes use of XMI/MOF specifications for defining the languages and the models, for this reason the PIM Editor is an open environment able to interchange models from external sources.

The PSM Editor is currently (?) Java-specific and it applies some DBE-specific technologies and frameworks, but this environment is not the only way for the completion of the software artefacts supporting the service. The decoupling, enforced by the MDA approach, with the support of the Service Execution Environment, provides a separation layer that allows the realisation of the services with a different platform. In the long term the project aims to support, for example, .Net services. The Integrated PSM Editor could be replaced by the SMEs with another IDE, based on different platforms.

In figure3 - Service Factory Generation Path below, the service construction path is depicted as arrows that move from the left. The normative approach in the service specification and realisation is to start from the CIM phase, supported by the Integrated CIM Editor: the Integrated PIM Editor contributes then to the specification from a Computationally dependent viewpoint. However, the process can also start from the PIM phase, but in this way the user will lose all the powerful business specifications that the reasoning and the recommendation processes have the ability to leverage.

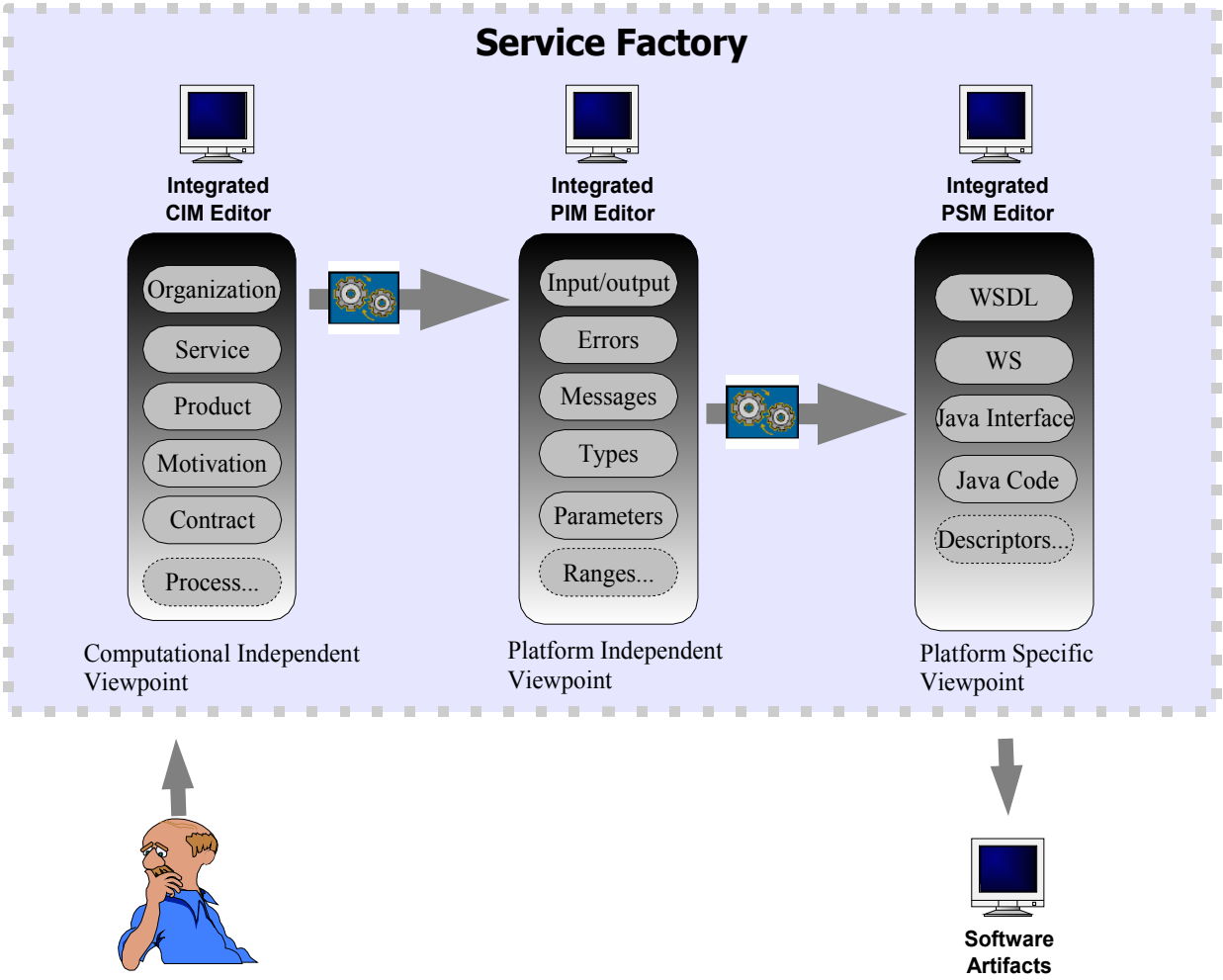


Figure 4 - Service Factory Generation Path

Languages and metamodels

The languages used to represent models in the DBE are essentially four[6]:

1. Business Organization metamodel (BOM)
2. Semantic Service Language (SSL)
3. Service Description Language (SDL)
4. Ontology Definition Metamodel (ODM)

The above languages are depicted in Figure 5 - Dependencies between languages below. The languages are separated in CIM, PIM and PSM. The Business Modelling Language is made of two languages: BOM and SSL. The first is related to the business representation from an organisational point of view, while the second represents the service from an interaction and business perspective. The OFM supports both a domain and an information system ontology; while the first represent the vocabulary in pure business terms (payment, currency, time,...), the latter is related to the information systems (string, text, range, types...).

6 In the DBE Consortium, TUC partner, has defined also a Query Metamodel Language (QML) for expressing MOF model based query expressions [QML]

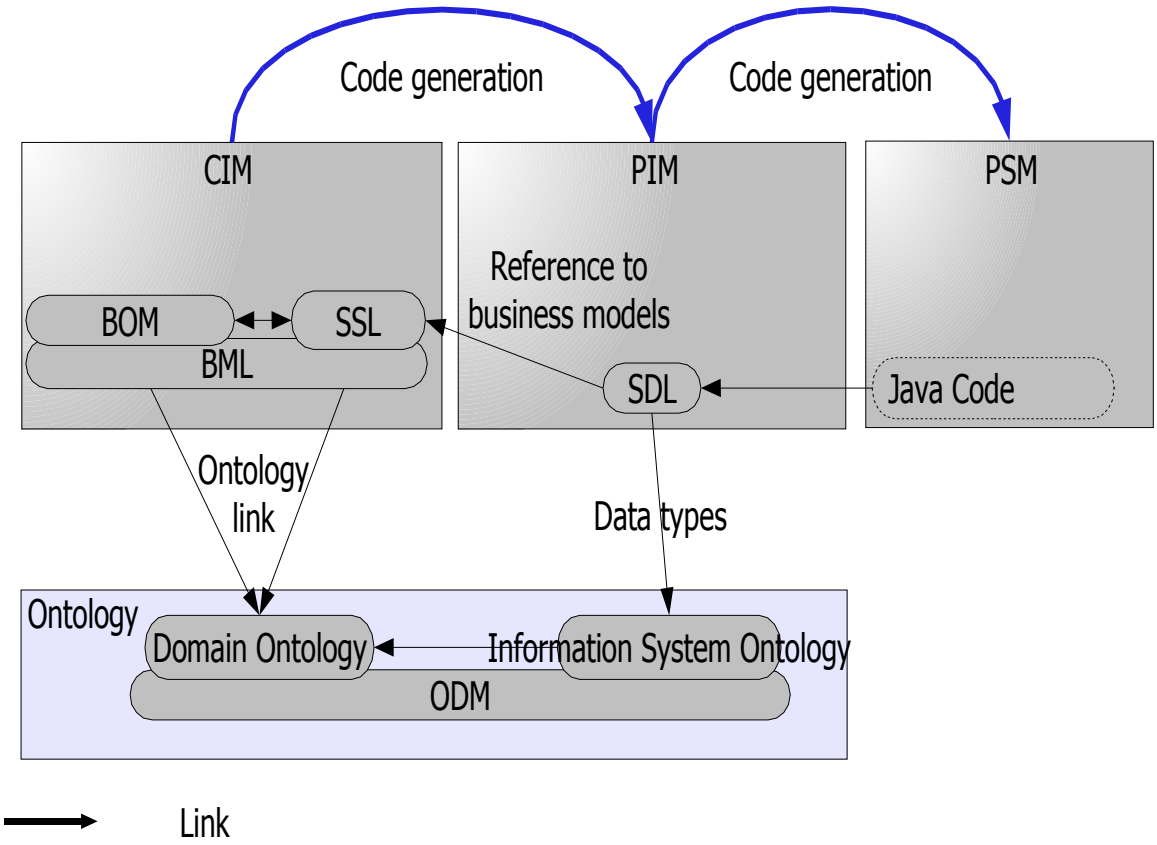


Figure 5 - Dependencies between languages

BML

The BML aims at supporting the description of business organisation, process, location, motivation and event which are the main questions addressed in the Zachman Framework [ZIFA87].

The SSL is part of the CIM phase that is focused on describing the business services from the interaction perspective. As such it puts in evidence in its constructs that a service needs and what it provides. Such specification does not aim to be a simplification of a computing interface and hence it has in most cases a close relationship with a service as defined in the BML. The BML "hotel reservation service" will correspond, for example, to "make reservation" and "cancel reservation", as a customer needs to be aware of when consuming the service. Structural services like "registerCustomer", "getTypeOfRooms" will not be modelled. For the same reason concepts like transaction, exception, parameters are not supported either.

The computable form of the business models can provide the ability to create on top processed to enable automatic contract negotiations and agreement creation, another goal is to compute the convenience of business offers.

SSL

Given an SSL model, we expect that there are many possible ways to specify the corresponding PIM. As stated in this document and in the "DBE Architecture Requirements", the project is not going to enforce some business and computing reference models as we might have done if we had adopted RosettaNet or ebXML. Rather,

the DBE allows each SME to express its models in whatever way they want. In order to address interoperability and mutual comprehension, it is fundamentally important to make use of an ontology system.

SDL

This SDL is a computational metamodel whose role is to describe the computational interface that supports the functional specification of a service. The related SSL specification needs to be extended with specific XML data types and with ancillary operations in order to support a technical interface. Given the SSL "makeReservation" and "cancelReservation" for "Hotel Reservation System" we might have to add "getCustomer", "getAvailability", "getReservation", "getRoomCategories" and so forth. Beside the enrichment of non business operations the SDL needs to take care of exception declarations, transactions and security.

The PIM modelling is still a phase that is related to the service and requires no effort to describe any structural requirements that are related to the DBE underlying infrastructure. There is no need to take into consideration deployment, distribution, registration or publication. The SDL is similar to the WSDL, but it has the ability to keep links with business models and ontologies and it has no binding section because it is entirely defined as a PIM based language (a significant difference from WSDL is also that SDL is a MOF-based language).

The SDL needs to be connected with the associated BML model: this will help the recommendation process to investigate the business organisation specification in order to provide the most suitable and adequate results.

ODM

In the Fetish project, mentioned in the introduction of the document, there has been intense use of the OPAL metamodel (Object, Process, Actor Modelling Language) [OPAL] developed by the IASI CNR. With OPAL, concepts are categorised by associating with computational independent entities, in particular, there are three primary types;

- Actor: an entity that is able to activate or perform a process;
- Object: a passive entity on which a process can operate;
- Process: an activity aimed at the satisfaction of a goal[OPAL2].

Other secondary information is available as well but not reported here for brevity.

The concepts in OPAL are linked together by two kinds of semantic relations:

- aggregation (called Vertical)
- similarities (called Horizontal)

The first comprises broader, part of and instance of while the latter describes Similarity, Predication and the generic Dependency within other objects or processes. It is evident that the semantic specifications are solely business oriented and there are no computing terms or concepts to be modelled that are outside of a business domain.

Despite that fact that OPAL is a powerful approach to ontology modelling that makes a clear cut between non Computing and Platform models, we have decided not to leverage it in the DBE essentially because there are no ontologies out there (except the one realised

by FETISH on the tourism domain) and because is has not been enrolled by any standard body on Ontologies; as a matter of fact the first obstacle is due to the second one. The current efforts in ontology modelling focus on adopting RDF/DAM-OIL/OWL and it is sustained by the W3C. This strong commitment in the community has the consequence of having ontologies that are widely used, like OTA. In the DBE we decided to create an ontology metamodel that beside being MOF based is also OWL compatible: the Ontology Definition Metamodel (ODM). Since the Semantic Ontology community is addressing the ontology more from a platform oriented specification than from the business perspective like OPAL, we decided to maintain the separation between the two dimensions that we called:

- Business Ontology;
- Information System Ontology.

Languages for the PSM

For the time being the DBE has not yet made use of MOF based languages for the PSM. There is a Special Interest Group (SIC) in the OMG called "MOF to text". Nevertheless the PSM might not even be useful in the DBE since, given the architectural specification, the transformation from PIM to Code might reduce the PSM to a documentation model.

The decision taken within the project consortium has been the use of Java as the platform implementation technology. The use of Java is not perceived as a constraint, due to the fact that it is possible to get a Java Virtual Machine for all platforms available in the market, in relative term Java is also "Platform Independent".

> **Note:** it is to be recalled that the DBE is essentially a Middleware and that actual business service implementation leaves outside the DBE infrastructures, namely in the SMEs own back-office. As a consequence the PSM does not refers to the actual implementation of the service, but rather to the specification of the service in DBE platform terms.

Code generation

One of the objectives of the DBE is to reduce the "Digital Divide". For this reason it is strategic to decrease the effort of the SMEs participating in the DBE. The long-term objective is to generate the code from the CIM specification. In order to achieve this it is required to have enough expression capabilities at the higher abstraction level in the MDA stack. In Table 1 - Type of code and generation capability below we summarise the types of code that are to be generated in a typical IT system.

When we have a model for the platform, the structural code, both the declarative and imperative part, is easy to be generated since it is invariant given a functional specification. Structural code generation is a well-known feature in IT (e.g. CORBA and RMI). It is important to note that with the current web service technology the structural code is about 85% of a classical ERP system. Obviously IT is moving forward in supporting the generation of this significant 85% since it reduces the time to market and decouples the dependency on the platform. The DBE is addressing this aspect, and provides the ability to generate all the needed code from the PIM.

The functional code is, on the other hand, far more complex and in most of the MDA solutions it is written in the PIM itself using an action language. One of the main goals of the DBE in the Service Factory is not just to speed up the code generation, but to decrease the SME effort, and this requires to get rid of any Platform specification and model.

CIM to PIM	**Declarative**	**Imperative**
Functional Code	50%	0%
Structural Code	-	-
PIM to PSM	**Declarative**	**Imperative**
Functional Code	100%	0%
Structural Code	100%	100%

Table 1 - Type of code and generation capability

As a consequence the CIM has to provide action languages or the ability to model business rules. Business models, processes and services need not just to be declared but must also be described from the behavioural point of view.

Until we tackle the research issue of how to generate PIM from CIM, the first approach will be to link an SDL model to the related BML models. A half-way home may be to generate a tentative service structure from the SSL.

The Service Manifest

The most important result of a working session with the Integrated CIM and PIM Editors in the SFE is the Service Manifest (SM). The Service manifest is conceived to represent a specific offered service from both the CIM and the PIM perspective; as such, it contains the models it is constituted of and the instance data.

The working session with the Integrated PSM Editor will provide the proxy that will mainly act as the platform-specific mediator to the remote back-end of the service. Given the fact that the SM is strictly PIM, the proxy is not part of the SM.

In more detail the SM it is made of:

1. CIM model
2. PIM model
3. instance data
4. quality of service data

The two kind of models that represent a service are called "Service DNA". This definition aims to enforce the distinction between definition (the models) and implementation (the data). The goal is to enforce the reuse of service DNAs by SMEs. This approach will ease the interoperability issues, more easily create a convergence on reference models, avoid fragmentation, improve reuse and speed up the realisation of services.

The instance data are built from the models and represent the information of the actual SME and service that is going to be delivered and offered.

Three features are the most relevant:

- it is computationally independent
- it is self-contained
- it embeds models as well as data.

This approach extends and leverages the regular web service technology where only technical information are provided in a consistent manner and where business and semantic data are published in an unstructured format. The service manifest is conceived to be completely PIM, except for the DBE metamodels (which are XMI/MOF based and hence transformable in anything that is considered to be useful from a computing perspective) there is nothing that forces SMEs to enroll new technologies or frameworks.

The SM, as detailed in the document "SM Conceptual Model" [STU04] and "SM Software Model" [SOL] represents a service offer in a specific moment in time and it is used to nail down a specific service status when defining a business agreement, as shown in figure 6 below.

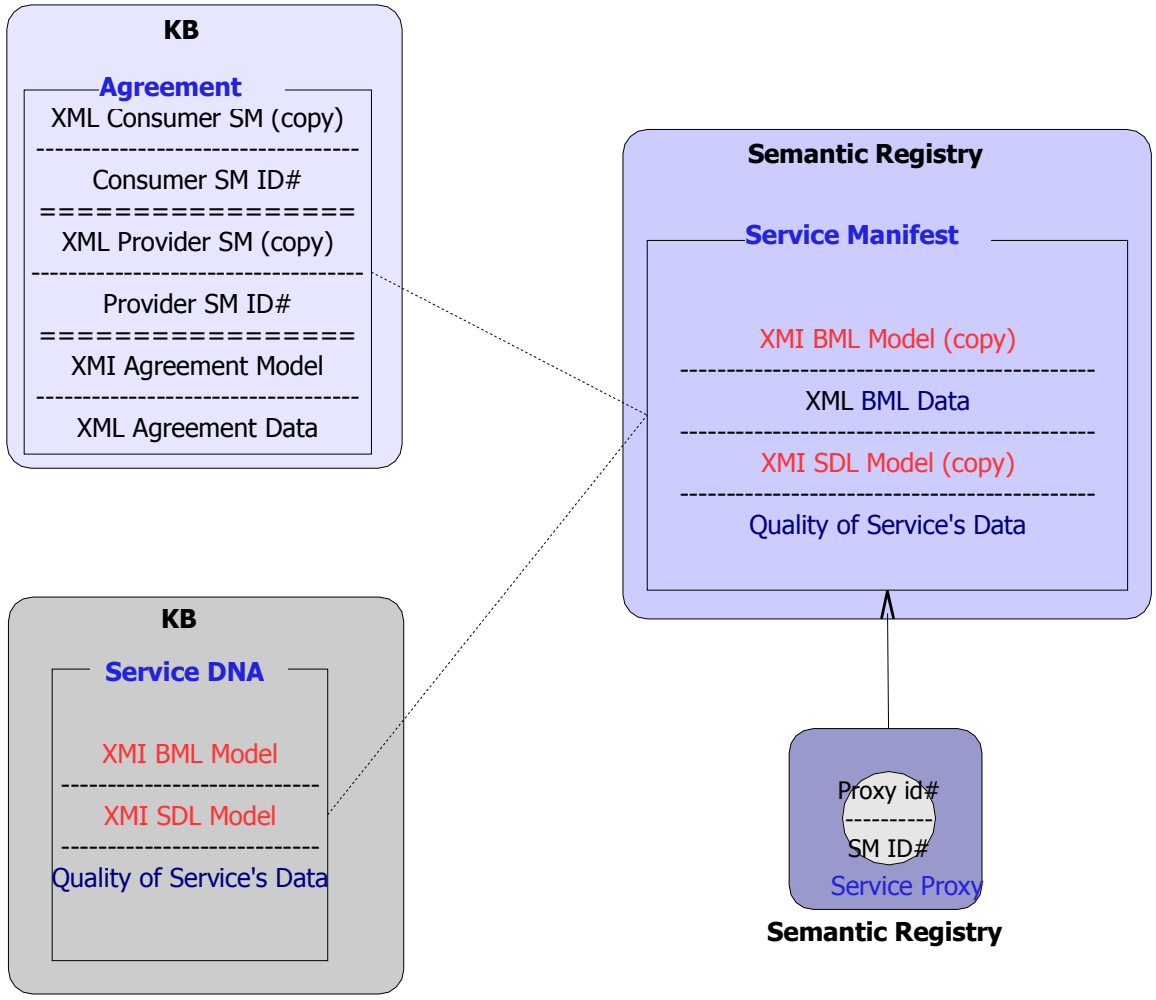


Figure 6 - Service Manifest, agreement and Proxy

The DBE Studio

The DBE Studio is the name of the visual environment where the various Editors of the Service Factory are integrated. It provides support for downloading updates, connecting to

ExE, and for providing side-support tools like validations, deployment support and so forth. The many tools are integrated and appear as a unique instrument to the user. The many languages used in the lower layers are invisible at all times. It essentially aggregates and links the three integrated tools (CIM Editor, PIM editor and the PSM Editor) and other features like:

- Service Manifest assembler
- Service registration and de-registration
- User logging and registration
- Support for code generation
- Service browser, model browser

The goal is to provide a seamless instrument that:

- orchestrates and integrates the various tools: from the user perspective most of the tools are not directly visible and the working session is mostly perceived as a path toward the definition of services.
- provides the most adequate User Interface for each SME's role: given the role associated to the logged user, the Editor will profile itself in order to show features and information conforming to the actual type of working session. For example it hides details about XML data types repository when modelling service from the CIM perspective.
- Acts as a bridge to the Service Execution Environment. Services will be deployed to the Execution Environment.
- Supports quick & easy updating of the tools: an integrated feature will provide a central aggregation point where the various tools are updated.

In Figure 7 below it is shown that basically the DBE Studio harmonises the various tools that feed and retrieve models from the KB Model Repository and provides Service Manifests and Service Proxies.

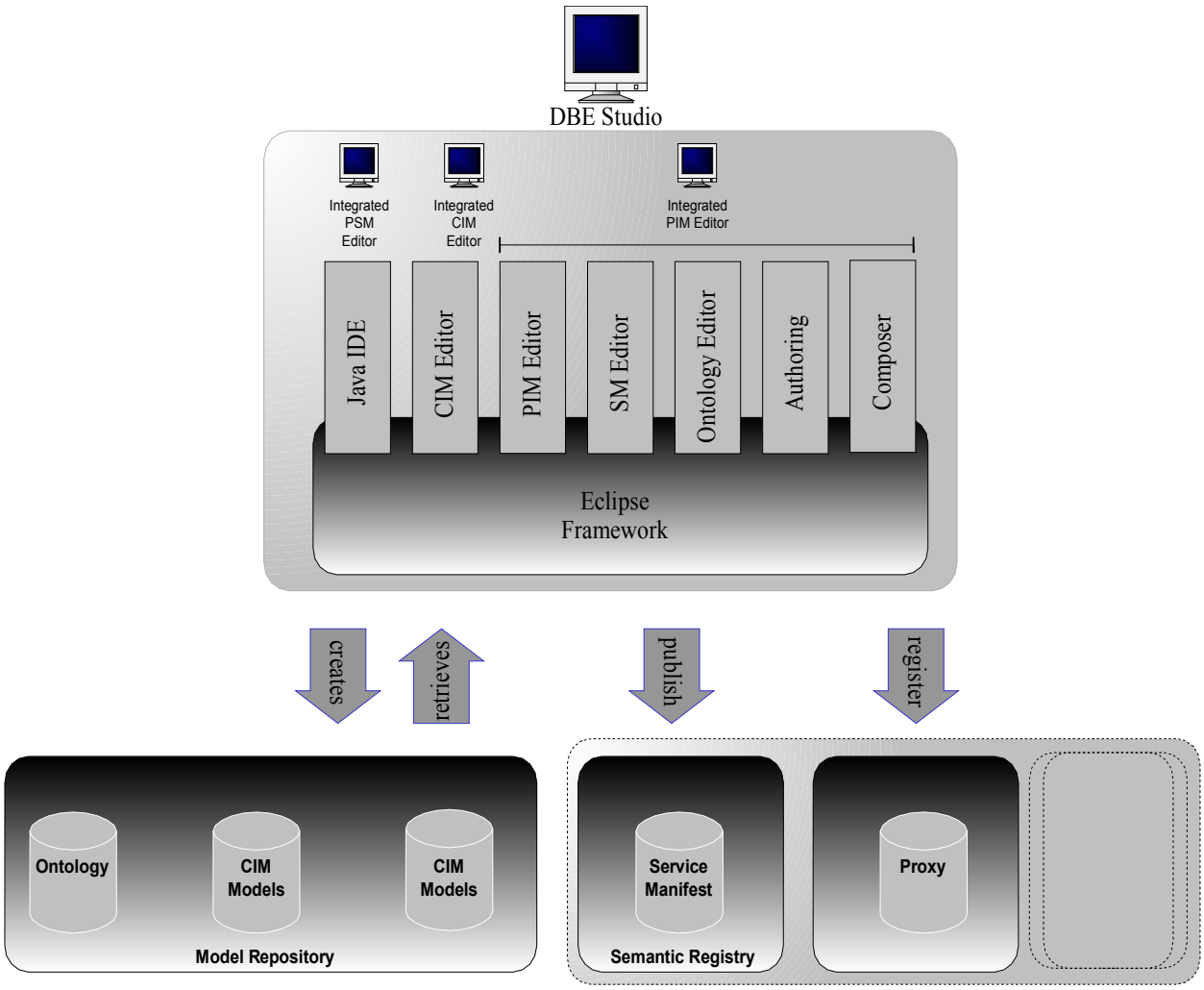


Figure 7 - DBE Studio dependency flow

5.2 Service Factory main Use Cases

Figure 8 below shows the main use cases that makes up the SF. They are grouped by the underling component that owns them;

1. BML Editor
2. SDL Editor
3. Deployment
4. Model Repository
5. Development Environments
6. Authoring Tool.

The Use Case have been organized in two diagrams shown in figure 9 and 10 below below. The first represent UCs that help to create service definitions, (i.e. to create the models), the second are related to creating an actual instance; i.e. the creation of the specific data for the SME and its publication

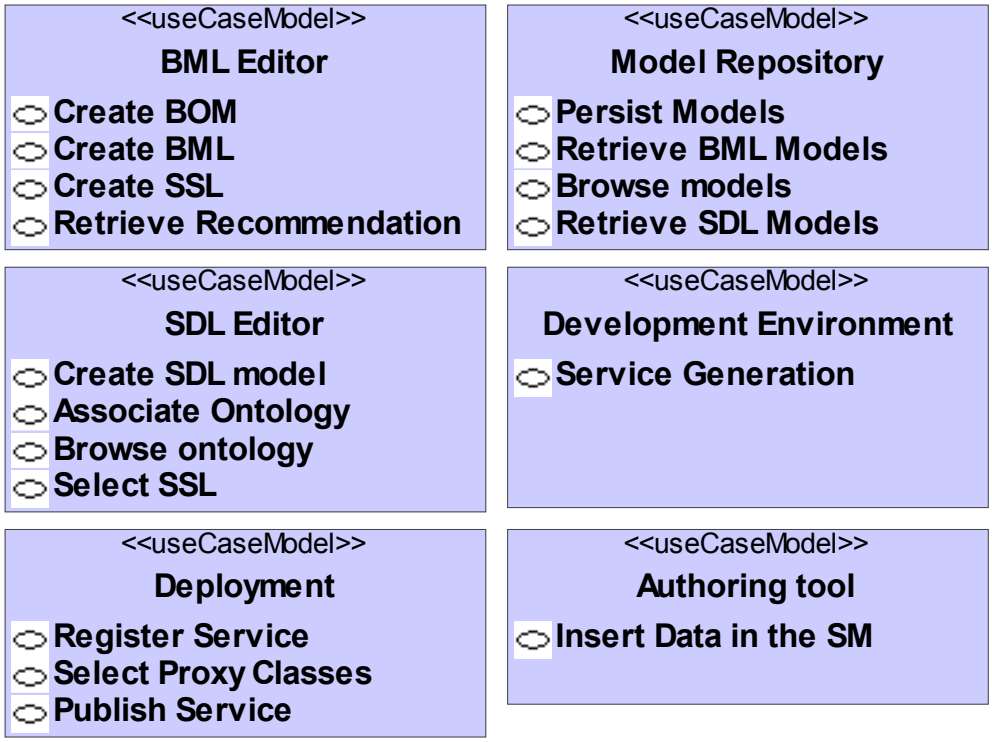


Figure 8 - Main Use Case Models

BML Editor

Create BML: A set of interfaces for creating a BML model. The model can be written from scratch or, as an extension Use Case, retrieved from the Repository. In addition a ready to be used model can be requested from of the Recommendation process.

> **Extension point** [search for service]: the business actor can ask the system recommender for pre existing and adequate BMI models
>
> **Extension point** [reuse models]: The business actor can decide to reuse already existing models

Create BOM: The Business Organizational Model is a supplementary metamodel in the BML and serves the goal of defining the SME specification from a business perspective.

Create SSL: The SSL is a separate metamodel in the BML, and serves the goals of defining a specification of the underling service from a business perspective.

Retrieve Recommendation: The recommendation, given a business request, returns a ranked list of BML models.

SDL Editor

Create SDL Model: It provides a set of UIs and facilities to create the SDL that has to be associated with a business model. It can be written from scratch or generated from the SSL model [extension: automatic generation]. It can be also defined as an extension or modification of an existing model [extension: reuse models].

For enriching the model semantically it is possible to associate an ontology [extension: ontology mapping]

Extension point [ontology mapping]: the users might decide to enrich the model by associating a semantic tag or vocabulary. This is an optional step, the drawback is that the service is less comprehensible by the recommender and related business intelligence. The advantage is that it is much simpler to create the model.

Extension point [automatic generation]: instead of creating the model from scratch it is possible to generate a structure from the SSL.

Extension point [reuse models]: it is also possible to try to reuse and adapt already existing models that can be found in the Model Repository. This is a suggested approach before creating the SDL from scratch.

Extension point [search for services]:

Associate Ontology: An ontology concept is associated

Select SSL: An SSL is selected from the Model Repository

Browse Ontology: The ontology is searched and browsed for the corresponding concept.

Model Repository

Retrieve BML Model:A BML model is searched and retrieved

Persist Model: Models are persisted

Retrieve SDL Model: A SDL model is searched and retrieved

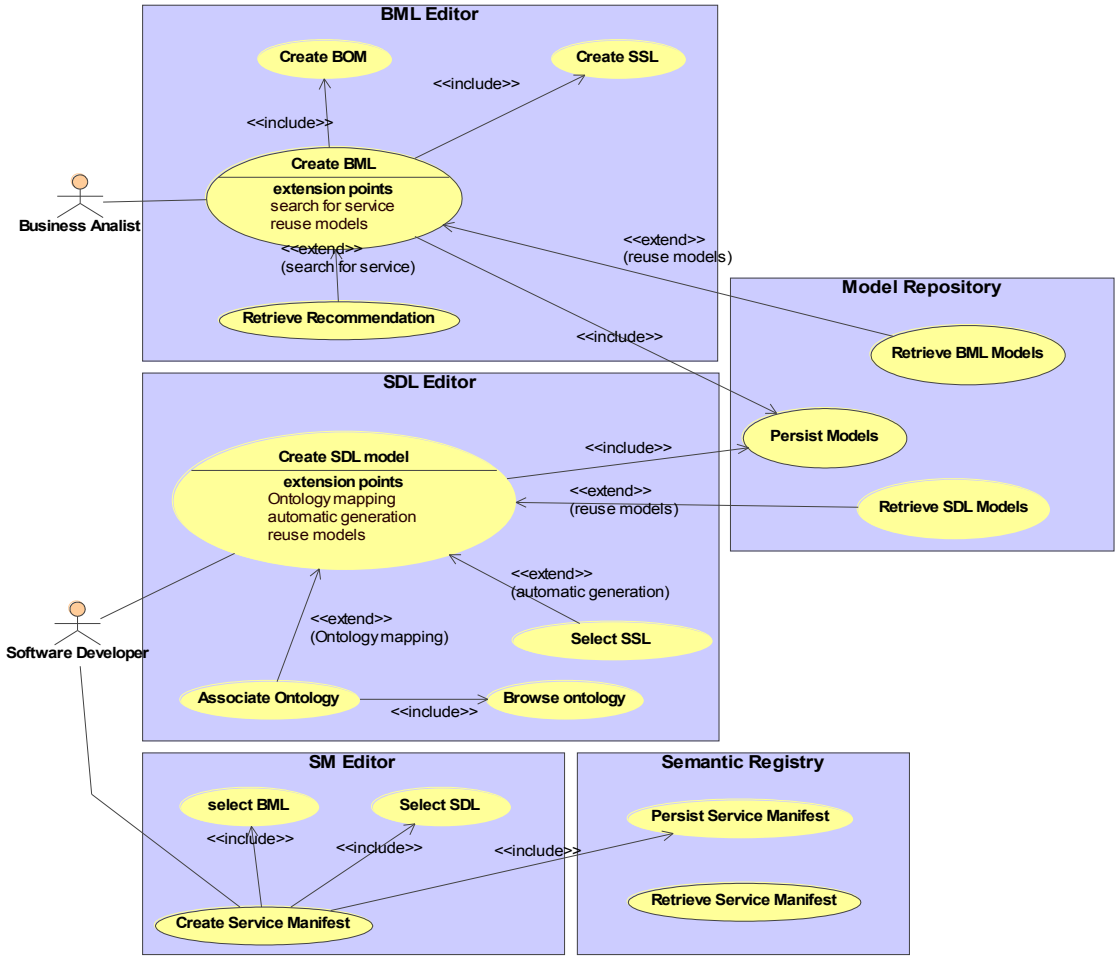


Figure 9 - Use Cases pertaining to the PIM viewpoint

SM Editor

Create Service Manifest: A Service Manifest definition is created as a composition of BML and SDL.

Select BML: A BML model is searched and retrieved

Select SDL: A SDL model is searched and retrieved

Semantic Registry

Persist Service Manifest: The Service Manifest is persisted

Retrieve Service Manifest: A Service Manifest is searched and retrieved from the Semantic Registry

These following set of use case, shown in the diagram of figure 10 below, are related to the publication phase of services.

Development Environment

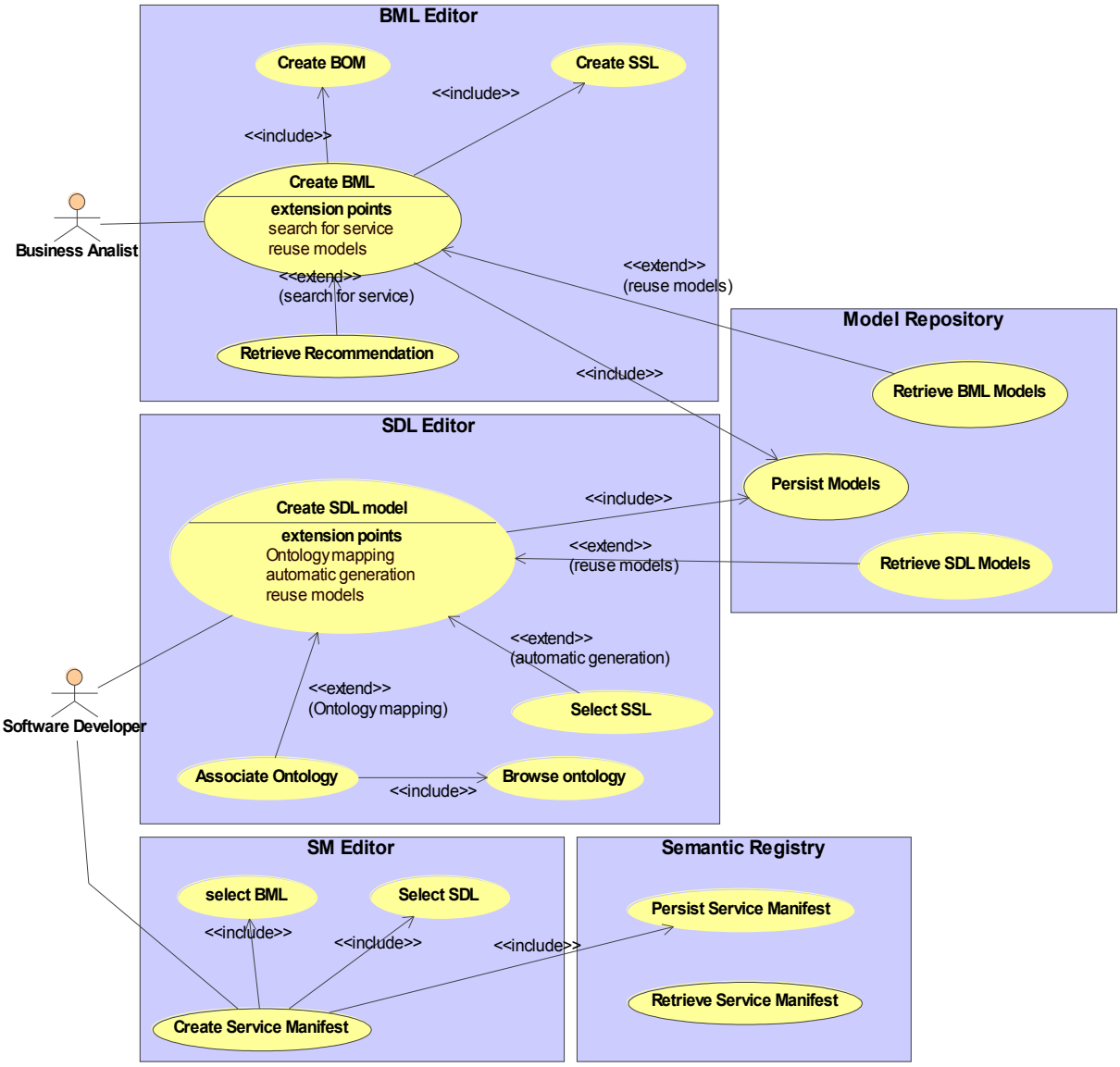


Figure 10 - Use Case Diagram for the model editors and the model repository

Service Generation: the DBE Java infrastructural imperative code of the service is generated. In the automatic/default way all the code is generated automatically -as much as possible- from the Service Manifest.

Extension Point [Custom Development]: The SME developer has decided to use a custom implementation. In this case the Java development environment provided by Eclipse is used together with the DBE frameworks.

Generate proxy: the code that actually invokes the remote service (the proxy) is generated automatically from the SM. The proxy as generated is just a mediator to the remote end-point where the service is actually implemented.

Proxy Development: the proxy is manually developed in Java without making use of the default mediator approach of the DBE. In this Use Case there is an almost total control on the service, hence it might be possible, for example, to implement the business behaviour in the proxy itself.

Deployment

Register Service: The service is to be registered in the Nervous System. In this way the service proxy is actually available to other SMEs and services. As a consequence a service is consumable.

Extension Point [Automatic Process]: the software developer has decided for the custom proxy and hence the actual code has to be selected

Select Proxy Classes: The Java class files that makes up the custom proxy is to be selected from the file system.

Publish Service: The Service Manifest is to be published. The service will become visible but not consumable, this happens when the SM is actually registered. This operation impacts only the Service Manifest.

Authoring tool

Insert Data in the SM: Information and data are added to the SM.

5.3Execution Environment

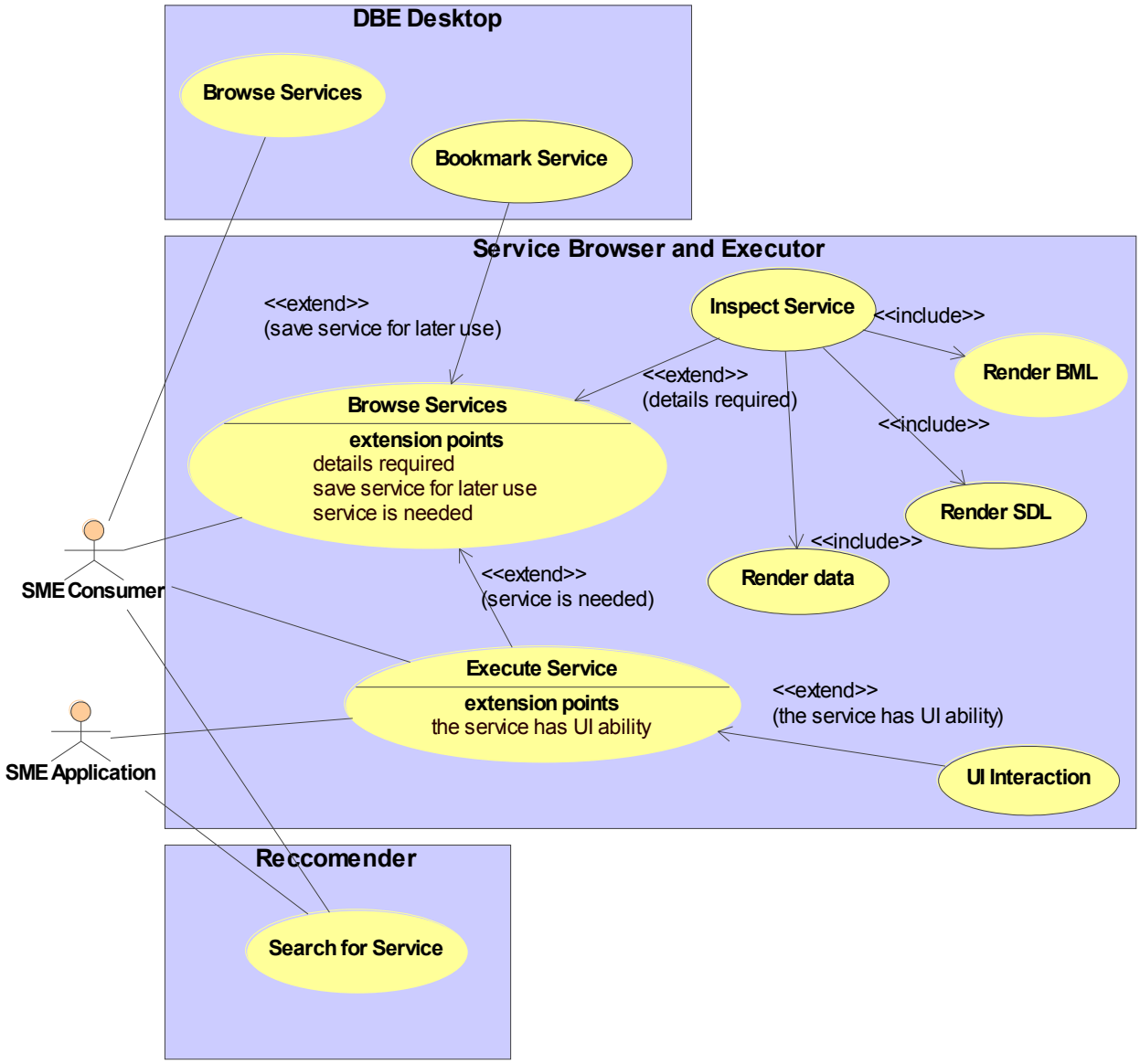


Figure 11 - Main ExE Use Cases

Service Browser and Executor

Browse Services: The published services are browsed, this can happen by navigating by business domain, publisher or simply by its name. This UC does not provide any intelligent recommendation, it is intended to work like an Internet Browser.

Extension Point [details required]: If more information about the service is required it is possible to introspect the service (i.e. the service manifest).

Extension Point [save service for later use]: The service can be saved in the user workspace in order to be executed later without the need to search for it again.

Extension Point [service is needed]: The service meets the user needs, and it is required to executed it.

Inspect Service: The Service Manifest is introspected, and all the models and data contained are shown

Render BML: The BML model is rendered and shown to the user

Render SDL: The SDL model is rendered and shown to the user

Render data: The data provided in the SM is rendered and shown to the user

Execute Service: The Service is executed

Extension Point [the service has UI ability]: the service has been developed with a UI and the consumer is not a process, for this reason a UI needs to be added

UI Interaction: The UI of the Service is create in order to consume the service interactively

Recommender

Search for Service: the recommendation process is required to suggest the most appropriate and convenient service to the user given the business and, optionally, the computational requirements.

DBE Desktop

Browse Services: The services stored in the DBE Desktop are browsed.

Bookmark Service: A service is stored in a local storage area for later retrieval for successive executions. This operation allows the user to create a library of services to fulfil requirements.

5.4 Evolutionary Environment and Distributed Intelligence System

The EvE provides the infrastructure for Distributed Evolutionary Computing to be used in automatic service-chain recombination and optimisation. This is a very significant challenge, due to the range of services that must be catered for and the potentially huge number of factors that must be considered for creating an applicable fitness function. Fundamentally the Evolutionary Environment will provide a computationally feasible interpretation of the models created from the science stream.

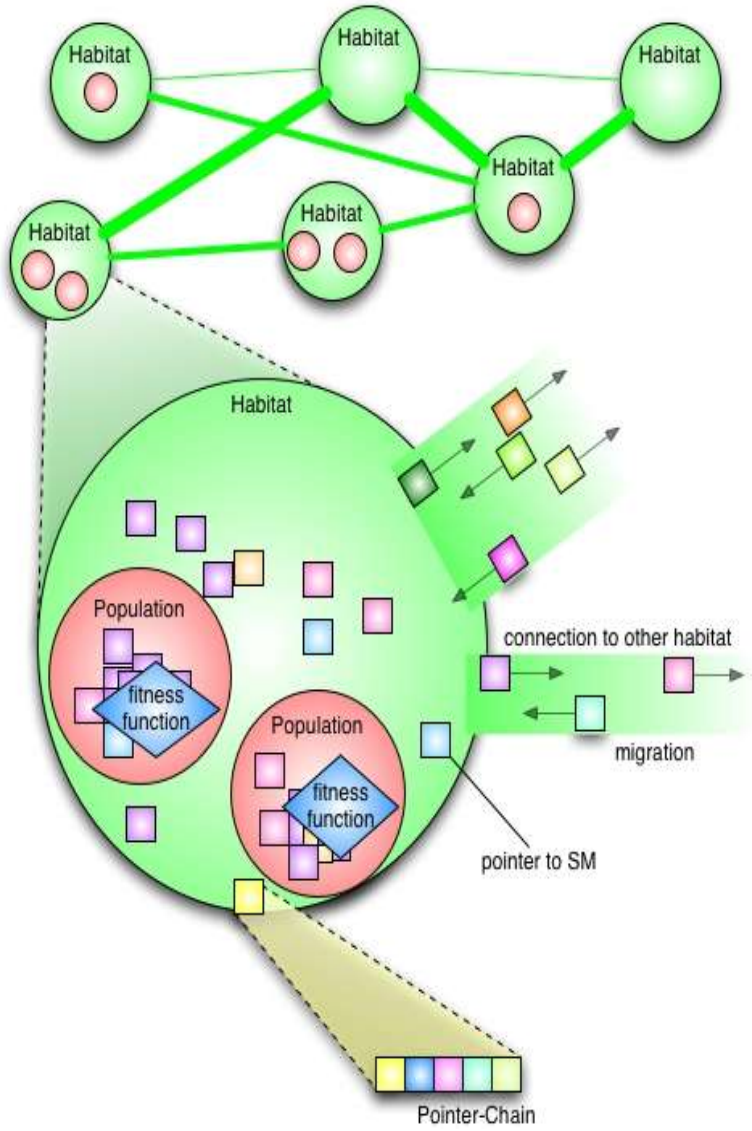


Figure 12 - Logical view of Evolutionary Environment (from [EvE])

At a high level the aims of the EvE are as follows:

1. For each SME user to create a subset of all the available services which is relevant to that user. This is called the 'service pool'. This concept is the 'distributed' element of the EvE.

2. Use evolution on the SME specific 'service pool' subset of services to create optimal service-chains in response to user requests. This is the 'evolutionary computing' element of the EvE.

The EvE will only operate on pointers to SMs, and not SMs themselves. Individuals in the EvE will be an ordered set of pointers to the SMs in a SM-chain. The SM-chain being the DNA of the individual.

Evolution can only be applied to a population of organisms/services, which of course over generations affects the individuals of the population. Replication and mutation are required so that change can be introduced in a population. The 'selection pressure' provides direction to change, and implies the death of individuals. The 'pressure' selects for those that are 'fit' and capable of surviving the environment to reproduce, and against those that do not have sufficient 'fitness' and die before passing on their 'genes'. Fitness is a measure of an organism's success in an environment. 'Genes' are the functional unit in biological evolution; in the DBE evolution the functional unit is the service. The 'genes' of an organism are strung together in a linear manner; the genes describe an organism's potential, its genotype. They therefore describe an organism's structure and behaviour in the environment, when it is instantiated (phenotype) at runtime.

For evolution to occur the following prerequisites are required; a population, replication, mutation, and a selection pressure. These will be provided in the Population component, which will provide a population initialised from the SME's specific 'service pool', the processes of replication, mutation and crossover. The selection pressure will be provided by a 'fitness function' which will be instantiated based on the SME user profile and SME user request.

In biological terms an ecosystem is a natural unit made up of living and non-living components whose interactions give rise to a stable, self-perpetuating system. It is made up of one or more communities of organisms, consisting of populations existing in their microhabitats.

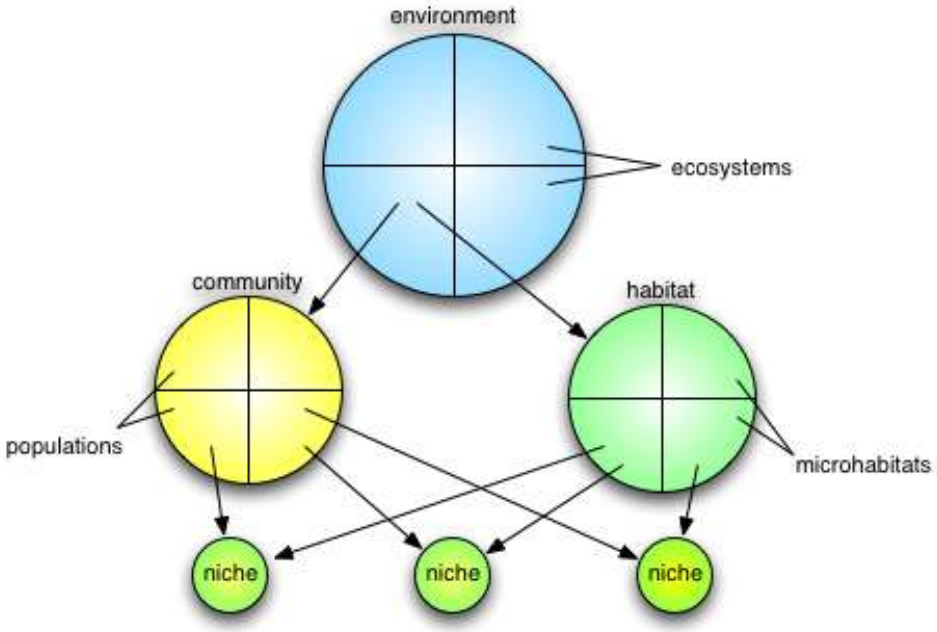


Figure 13 - Abstract Ecosystem Definition (from [EvE])

Evolution is a very specialised task, which needs to be localised to specific habitats (SMEs) and populations (requests), which leads to the creation of niches. There also needs to be support for the SME specific 'service pools' from which solutions are to be constructed. This functionality will be provided by the Habitat component. Each SME will have a Habitat initialised with their SME profile to know which services in the DBE are relevant to the SME it represents. The Habitats will be interconnected, and as pointers to the SM migrate across the EvE through the habitat communities, the SME Habitat service pools will increase in sizes with services relevant to the SMEs that they represent. Allowing a small amount of migration of highly fit individuals can greatly optimise the evolutionary processes. Empirically, this seems to work well-though there is little theory[7]. The Habitat component supports 'distributed' evolution which is required to create an ecosystem as understood in biological terms.

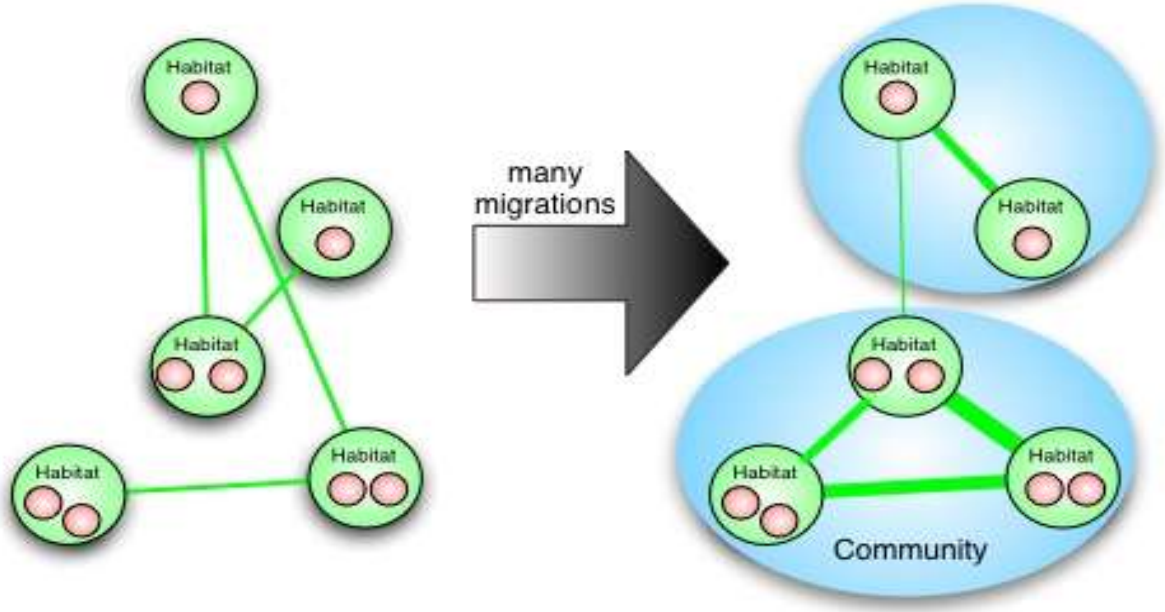


Figure 14 - Community Formation - Habitat Clustering by Service Migration (from [EvE])

The connections between the habitats will be initialised randomly at first and later based on the SME profiles. These connections will then be updated based on the 'usage histories' of the migrating services and will allow the spontaneous formation of communities via habitat clustering. Many successful service migrations will reinforce habitat connections, increasing the rate of service migration. If successful migration occurs due to a multi-hop migration, then a new direct link can be formed between those habitats. Unsuccessful migrations will result in connections (rate of migration) decreasing, until finally the connection is closed. This reinforcement of links is similar to pheromone trails and is a memory feature that leads to the self-organisation of the communities.

This community clustering is like joining a club, you can be member of more than one. In the same way habitats can be clustered into more than one community. The clustering will lead to communities forming based on language, opportunity spaces, nationality, geography, etc.

The Distributed Intelligence System (DIS) will provide an accelerator to the evolutionary process in a form of a Clustering Catalyst for the Population component. The Clustering

7 A similar model has been used for determining investment portfolio strategies\cite – http://www.codefarm.com/news/item.jsp?id=15

Catalyst will use a neural network, at the the level of the Population component, for pattern recognition based intelligence to encourage the formation of clusters. Within these clusters preferential crossover (genetic recombination) of individuals within the clusters will be encouraged. This will accelerate the evolution of the perceived desired features. This works similarly to when humans use selective breeding in plants or animals for specific features and qualities, by enforcing recombination between individuals that encourage the desired features and qualities.

A preliminary figure of how the EvE integrates with the SF and the ExE can be found at the end of report D18.1.

5.5 Model Repository

The DBE model repository, a part of the KB, represents a serious shift in data management. In standard model repositories the structure is conceived to be a hierarchy of classes and entities in general. The information is supposed to be accurately managed and organised by an administrator or by a small group of expert users. In such repository the common approach is to browse the tree-view to select and add new models or classes. It is assumed that each user knows where to look for the needed data and where to create new entries . These assumptions works fine in a working environment with a limited number of users and of data. Usually this happens inside a project team that concurrently works for the realisation on a single, although big, project. Nevertheless the team has to conform to a rather strict management process in order to support the coherent organisation of the information. Often the organisation resembles the model hierarchy itself. For example, for each component like accounting, financing, logistics and warehousing , defined by the architect or the business analyst, there will be a team in charge of enriching and maintaining it. Interoperability and coordination between components and the related models is performed by the architect that has an overall vision of the project.

It is not rare that complex projects are made of hundreds of components and of thousands of classes. In such cases the generated models are often difficult for a single person to understand. This happens despite the approach to software development (either extreme [XP] or RUP based[RUP]) and the software life-cycle enforced in the team (either waterfall or iterative). It does not help much to have a tree-view or a search mechanism, because the relational criteria that were used to model the structure might not be clear and consistent with the different mindsets of the users. Moreover, a search mechanism is usually not semantically driven but it simply matches tags wherever they are used.

In a context like the DBE where, referring once again to the Herzum-Sims Information Level Exhange [HS2002], the information exchange occurs at level III between SMEs and information modelling is done inside in an uncoordinated way relative to other SMEs. It is evident that the impossibility of having a unique supervising entity, a global administrator (if we reject the idea of having a "Digital God" in the DBE) will have the consequence that the regular project-like approach to model and information management is not applicable in the DBE. Moreover the regular tree-view approach to model browsing is not applicable with the incredibly large amount of information that the DBE is likely to create and manage (as stated several times in this document).

The DBE model repository has to provide a new approach to model management, to browsing and to searching. Semantically rich descriptions of models need to be provided,

together with the ability to show the various kind of dependencies in a smart way. A plain tree view cannot be applied essentially because, given the absence of a "DBE God", the natural organisation of models will be similar to a mesh rather than an ordered hierarchy. In the same way, for example, the structure of the Internet does not follow a precise topology (Ref. Figure 15 below[8]), since it is driven by local rationales like performance, local bandwidth optimisation and cheaper connections. It is because of the local optimisations and scope of participant SMEs that at "10,000 feet" the entire model does not show a hierarchical structure.

Note: Even though the image provided might contradict at first glance the initial assertion about the nature of Internet, the authors of the image (Bill Cheswick and Hal Burch of Lumeta Corporation) assert that it is a tree-like topology, i.e. the layout can be partitioned in subsets where a tree-structure can always be identified, but the entire map is indeed a closed mesh.

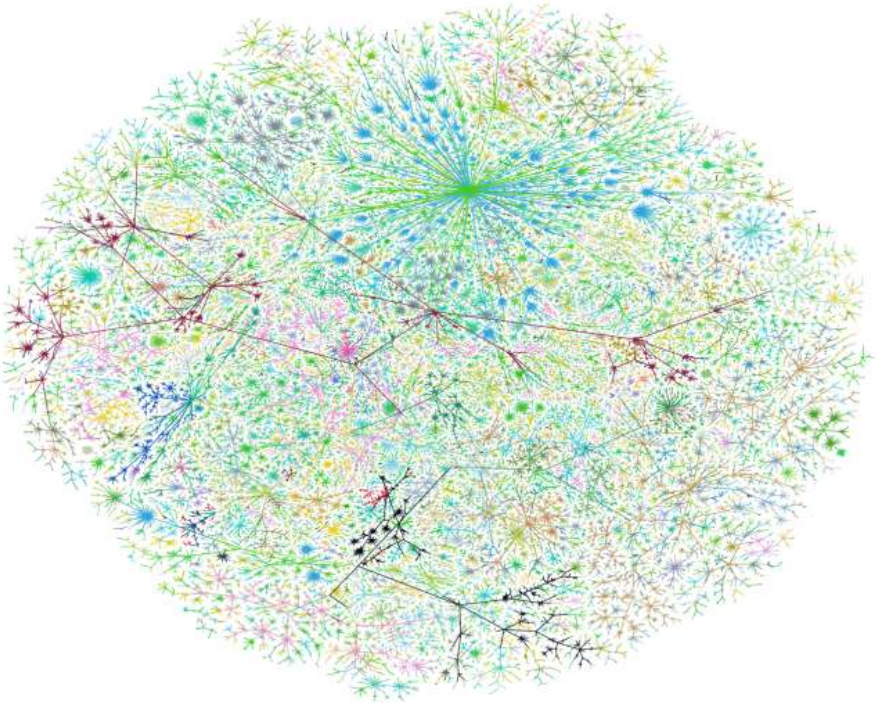


Figure 15 - "Internet Mapping" June 1999

The relationships between models created by SME are:

- "horizontal" when connecting SDL models, BML models and business data types and
- "vertical" when linking with other pre-existing models in the SME outer space: an SME has the option of reusing pre-existing models with "is-a", extensions or "part-of" relations.

In addition, every new model will be related to models from which it has inherited some characters, and this information is preserved in the Model Repository because it constitutes a valuable way to derive evolutionary information. This multidimensional connection space is valuable to maintain the status of the ecosystem from Model, Service DNA and Service Manifest viewpoints.

8 http://research.lumeta.com/ches/map/gallery/index.html, Internet Mapping Project, http://research.lumeta.com/ches/map/

Moreover, models will keep ontology references as well as business domain information. All this kind of information is stored and used for recommendation, for browsing, for filtering in or out the valuable information and for supporting the evolutionary environment (EvE). In general the Model Repository faces the challenge of implementing what we might call "model mining" in comparison to more common "data mining" with relational data bases.

Semantic distance

It has been said several times in this document and in "DBEArchitectureRequirements" that service interoperability is an issue mostly because there is not a "DBE reference model" that each SME has to comply with. With this assumption, PIMs of services are naturally incompatible from both the semantic and the functional viewpoints. Even though we expect that the direct support of the evolutionary features will allow services and SMEs to converge to common models, there is still the need to address the requirement of SME Consumers and SME software developers to understand if and how different PIMs of services are compatible; this issue is identified by the term "semantic distance". Essentially it means that two PIM models, despite the tags (aka naming conventions) and the message specification (operation, parameters and data type used), have a semantic distance. Named PIM A and PIM B the first and the second model, Fsd(PIM_A XMIModel, PIM_B XMIModel) returns a semantic distance real number (sd) that ranges from 0 to 1. The closer the sd is to 1, the more the two models are semantically equivalent, and in this case they are semantically and functionally compatible; i.e. They represent the same business service[9].

Computing the Fds() function is a problem that is addressed with the help of common ontologies (mind the plural), the ability to understand the relationships from the inherited models, the business context, usage information and any other model or relations that will be part of the model.

9 The semantic function is not relate to the Service Manifests but to models and Service DNA; the problem of identifying best fitting actual services is a problem of the Recommender, that in turn can make used of the semantic distance.

6 Technical Architecture

6.1 Service Factory

The desktop part of the Service Factory is built on top of the Eclipse Framework. It provides many useful features in the context of the DBE. Namely it supports the editing and processing of any languages thank to the adoption of EMF, a MOF based repository that supports model transformations and XMI processing features.

In order to address the need to avoid defining a single functional reference model and for supporting data types not known at development time, the DBE structural services uses XMI for interchanging models.

The KB and the SR adopts a P2P technology for providing a decentralised data storage. In addition the service interface adopted for model and metamodel interchange is JMI 1.1 compliant. The consortium will move to JMI2.0, supporting MOF2.0 as soon as it will be finalised.

The KB adopts a P2P technology for implementing a decentralised approach to data storage.

6.2 Service Execution Environment

The DBE Service Execution Environment (ExE) is where actual services live. They are registered, deployed, searched, retrieved and consumed. This parallel world is sometimes referred to as the "run-time of the DBE". As a consequence, the ExE is the gate to access the DBE services.

In fact, the ExE "is" the actual DBE Network. The DBE P2P Network is created on top of all the instances of the ExE. Each single instance of the ExE contributes with part of its resources to the whole DBE P2P system. This set of ExE instances behaves as a unique system with its own identification service (distributed identity), its own persistency service (distributed storage), etc...

On the other hand, the ExE is the bridge between technology and an important part of the results coming from science. As the ExE is the DBE application bus (nervous system), it is where all results regarding network behaviour, network topologies, stability and symmetries, self-healing, self-organisation, evolutionary environment, etc... have to be applied.

Even if, from the technical point of view, the ExE can support the entire implementation and behaviour of the services, it is best described as a substrate, a soil where SME services get connected and interact. This statement is in line with DBE as being at Level-3 in the ILE (reference "DBE Architecture Requirement"). The DBE is not supposed to replace the back-end infrastructure systems of SMEs, but rather to act as a wide smart Internet-enabled bus that enables services to interact, evolve and integrate. Moreover the ExE provides transparent integration with the Evolutionary Environment (Ref. "Evolutionary Environment") and with basic services like accounting, security, logging, payment and so forth. In figure 16 below the relationship between the two environments is shown.

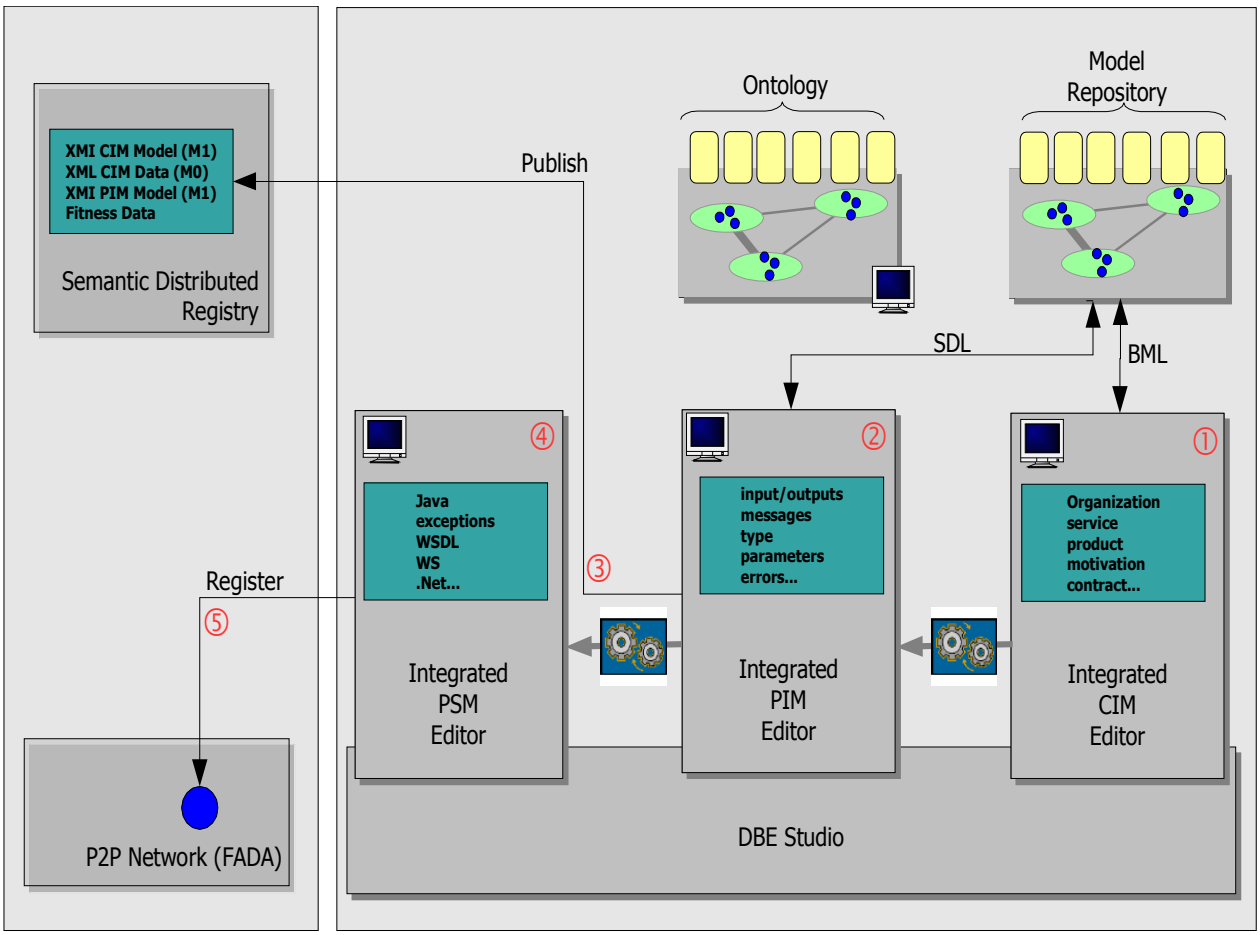


Figure 16 - ExE and SFE interoperability

In step 1, shown above, the BML model is created. At step number 2 the SDL is created and at step 3 the service manifest is published in the Semantic Registry. In step number 4 the service proxy is generated and at 5 it is registered in the P2P based Nervous System. Step 3 and 5 depends on the Execution Environment.

Service Registration and Deployment

Once a service has been defined in the Service Factory Environment, it must be implemented in order to be used. This implementation can take place in one of two main ways. The first one is by creating an implementation from scratch that complies with the SDL (WSDL) definition of the service performed in the SFE. Compliance with the business rules defined in the BML expression can not be validated for the moment.

The other is by writing a proxy that adapts an existing service (for example, a Web Service) to the semantics specified by the BML-SDL pair.

Once this service is coded and deployed in an SME, a proxy for the service must be created. This service proxy is then registered in the DBE Nervous System. The proxy is accompanied by the service manifest.

Another ExE goal is to hide to the upper layers (which means, to the regular programmer) all the complexity inherent in distributed systems. As a consequence, all services available in the whole DBE will be perceived (designed and written) as if they were deployed locally in the ExE, without the need of taking into account the interactions with remote instances of the ExE.

The main functional features of the ExE are summarised in the following paragraphs.

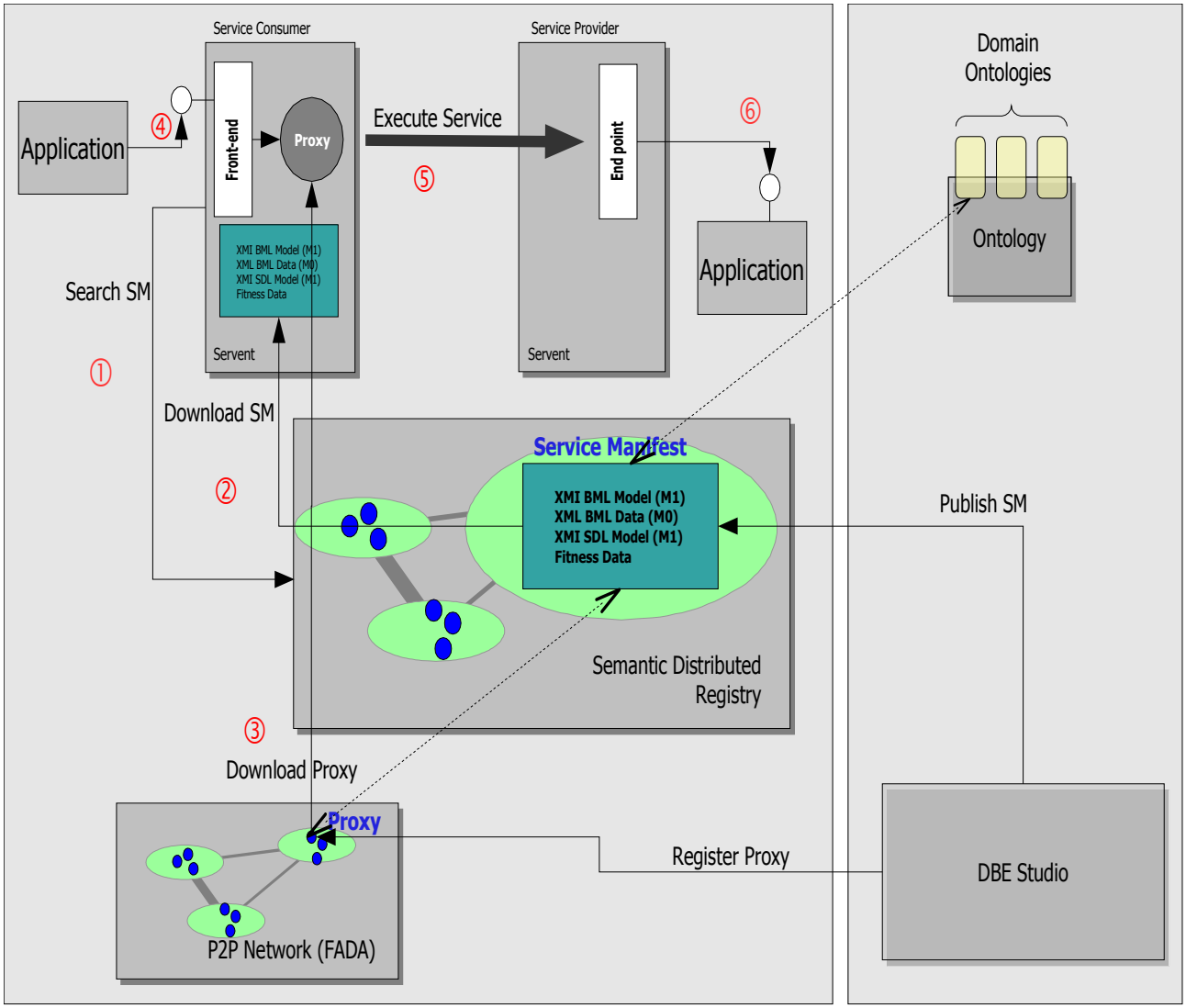


Figure 17 - Service Search, Retrieval and executon

Service Search, Retrieval and Execution

When one SME needs to execute a particular service it must first find which service suits its needs. To do so, the recommender tool is used. Given a set of criteria for the service needed, which include the SDL and BML expressions and possibly some M0 data such as the company name, the recommender provides a list of service manifests that match the query criteria. By selecting one of the Service Manifest, the related service proxy is downloaded in the DBE Servent Server of the client SME.

> Note: the Servent (SERVer & cliENT) is the piece of software which creates the DBE P2P network transparently and contains services (infrastructural and custom made). The services ready to be deployed in the Servent are created in the SFE.

Service Execution

Once the service proxy is in the Servent of the client SME, the client may execute it. The service proxy makes use of the UI framework that is installed along with the DBE

Application Server (called "Servent"). The result is that the service UI pops up in front of the user, who may start using it. The user may be guided through a series of pages in order to formulate the desired request. When the request is created, the service proxy will invoke the back-end service, through the services offered by the Servent. In this invocation the accounting system will be called from the Servent of the client SME, and also from the Servent of the provider SME. Both accounting steps take note of several parameters, such as the identity of the caller SME, the identity of the called SME, the service, the operation, and several other parameters that are needed by the accounting system to perform its goals. The results are returned along the reverse path, and pass through the accounting system as well. The results appear in the client SME UI.

These steps are described in the figure 17 above. Steps 1 and 2 happens in the SF, step 3 represent the search phase in the Service Manifest. This phase requires a proper UI (the SR browser and query builder) but it is shown in a simplified manner in this figure. In step number 4 a SM is chosen and hence downloaded. Prior to execution time the service proxy is downloaded from the Nervous System in the Servent and used to actually execute the service, that in this case is remotely implemented.

A more detailed description of the execution phase is provided in Figure 18 below where it is shown that the Servent is providing the ability to wrap the service proxy at runtime and provide a SOAP interface to the consumer legacy applications. In addition the Servent, taking advantage of the introspection ability of the proxy interface and specification to create a User Interface.

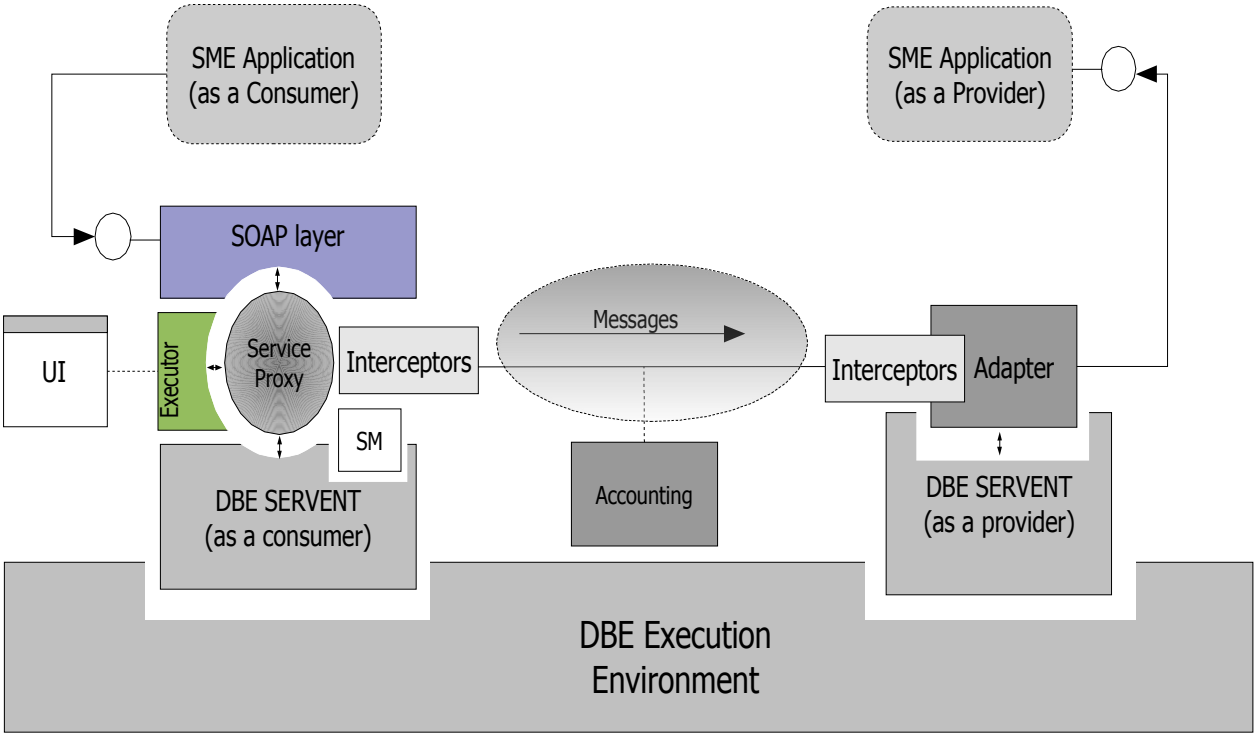


Figure 18 - Service Execution details

Nervous System, FADA

The middleware in the DBE that is in charge of storing and distributing the service proxies is called the DBE Nervous System. It is transparent to users, by means of the Servent, and the coordinated interaction among different infrastructure services provides the illusion of a single centralize service registry.

FADA (Federated Advanced Directory Architecture) is the name of the SUN's implementation technology, it is based on the SUN's Jini concept and it implements the Nervous System. The main features provided by FADA technology is the ability to be a decentralize storage area which maintains a link (lease) with the registrar process, in this way the service proxies are always in-sync with the process that provides access to the service. In addition FADA is able to work in the Internet and be tolerant to some extend of network and hardware failures.

APA

This is the Abstract Protocol Adapter, created inside the DBE project with the aim of wrapping application server APIs. This will ease the effort of using different types of Application Servers: at the time being the consortium is making use of the Apache Axis implementation of a WS engine.

6.3 Evolutionary Environment

The EvE (Evolutionary Environment) is transparent with respect to the other two environments. In order to allow the ExE and the SF not to be aware of the existence of the EvE and hence not to hard-code any EvE aware code a event based publishing and subscribe a message protocol has been implemented. Essentially the EvE will receive usage information from the ExE (see Figure 19 below) and send back the created service chains.

In more detail, as described in figure Figure 20 below, the EvE has some more dependencies because it has to access KB, the SR for gathering more information about the services. What is important to highlight is that the EvE only deals with the CIM part of services and it does not try to solve the PIM mapping. This will be addressed by the manual composer. This means that the only meaningful piece of information is the CIM model (BML and SSL), the manual composer will try to compose services also from the computing perspective, i.e. the by examining the SDL.

Figure 21 below draws the suggested network mapping and deployment schema. The approach is to create several instances of EvE, one for each habitat. This mechanism will reduce the network traffic and optimise the distributed storage mechanism.

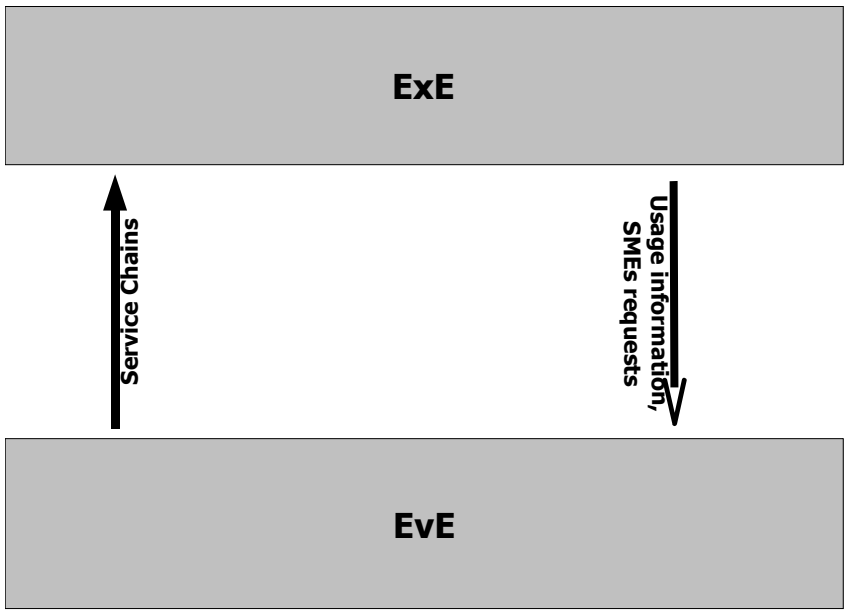


Figure 19 - EvE in relation with the ExE

In detail the dependency with the ExE is more complex, as shown in figure 20 below. The Servent's interceptor architecture sends usage information in the DBE long term memory implemented by the KB, the EvE retrieves the information and also gathers Service Manifests from the Semantic Registry. In order to enrich the semantics of the service it retreives also the ontology from the Service Factory; recall that SMs do not contain ontology data, but only a link.

Once new services have been created the new BML is to be published in the Model Repository. In case a service chain has been created, it is passed to the Automatic Composer supporting the user in creating the computational interface, which in turn is first published in the Repository and in the Semantic Registry. The proxy is not created because the onus of the EvE is to create service specifications, not to service implementation, which would also be rather impossible at the time being.

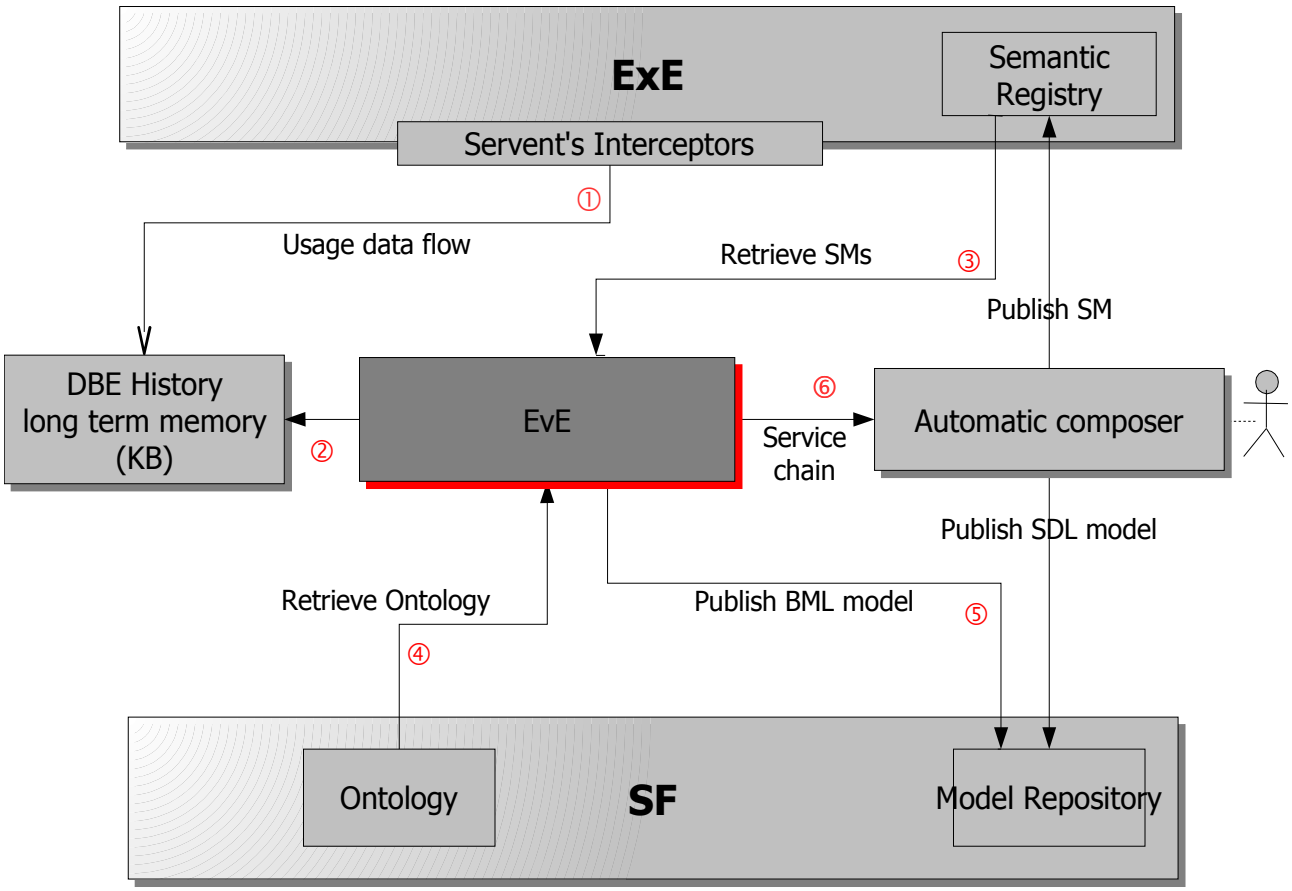


Figure 20 - EvE and its detailed dependencies in the DBE

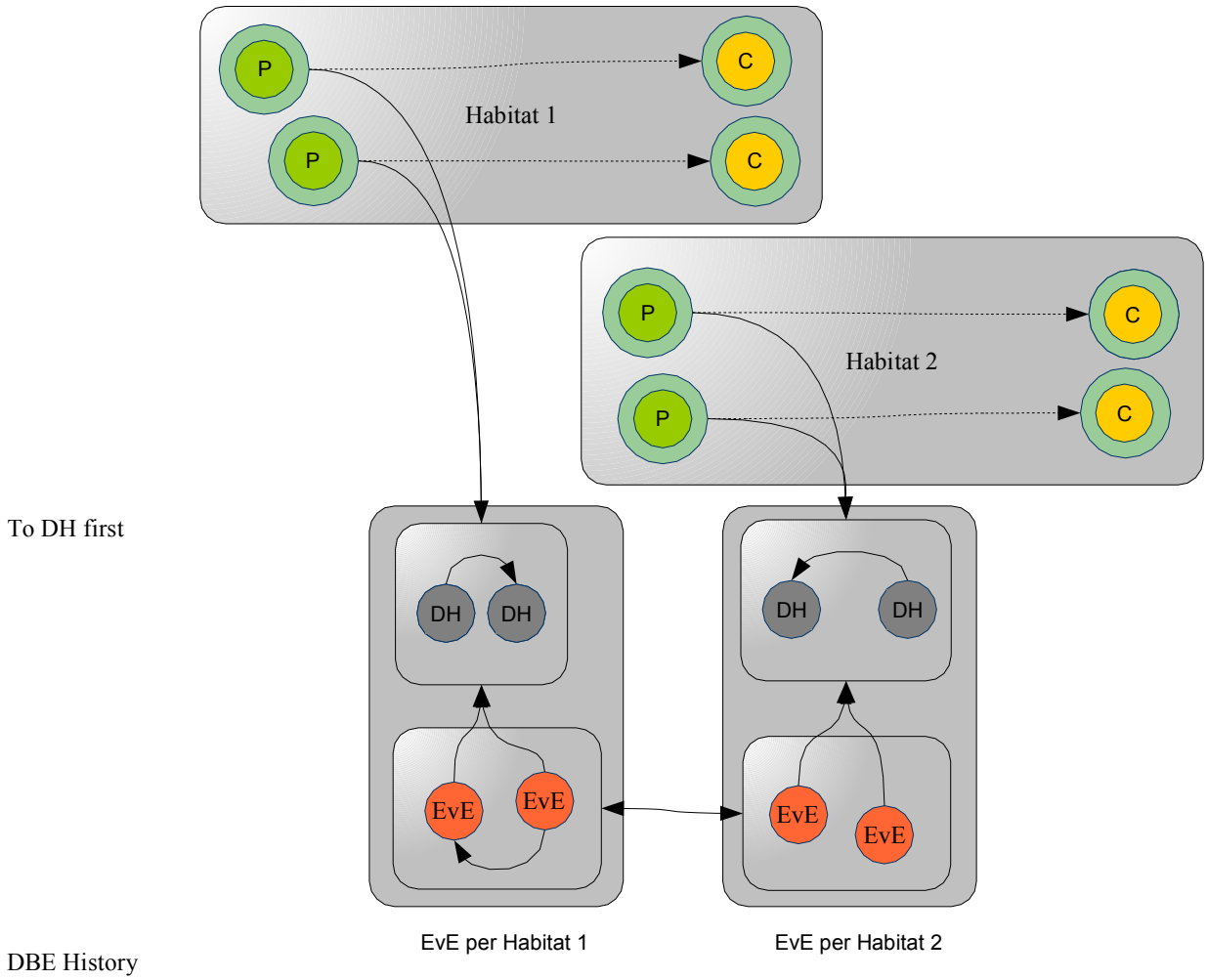
The Evolutionary environment is conceived and implemented as a decentralized system and is hence made of a cluster of Internet distributed EvE nodes. This approach guarantee that no one is in charge of controlling service evolution and that a failure in a single system makes the EvE disappear.

Implementation strategy

The ExE, as the KB and EvE, is a distributed and decentralize environment -- and hence it is made of a network of Servent nodes. There is the risk that the usage data flow messages from the ExE to the KB's Long Term Memory saturates the network bandwidth. In addition the EvE evolutionary processes has to search the Long Term Memory and this can consume some significant bandwidth. This intense flow of messages can jeopardize the performance of the DBE and saturate the available network capacity of SMEs.

In order to address this issue the implementation has to map the EvE vision of the SMEs community as a customer of habitats. Each habitat is hence to be realized as an interconnected set of EvE nodes, called "EvE per Habitat" (reference to figure 21 below). With this approach the message flow is limited to a single habitat and the evolutionary processes originating from EvE nodes does not spread across the entire network.

Specific processes for a global optimization can cross the EvE per Habitat but these are special cases that will not create much traffic.



DH: DBE History

Figure 21 - EvE and the network topology

6.4 External Interoperability

Although traditional evolutionary computing is widely used for optimisation problems, there are no standards for interoperability as it has only been used for stand-alone applications. The Evolutionary Environment is a form of distributed evolutionary computing (DEC) for which the potential is just beginning to emerge. Its use in general is rare, and is unique in the field of the DBE. So there are no existing standards in DEC to build upon. The effort of integrating with other Digital Ecosystems is at the time being an intellectual exercise also given the fact that other Digital Ecosystem are not available yet. Given that, we will simply focus on interoperability with the data and model viewpoint

The adoption of the MDA standard, namely MOF and XMI, allows may components of the DBE to import foreign models. The SDL editor can import and transform WSDL and Java interface files, the Model Repository can import OWLS ontologies, the Service Composer can import BPEL workflow descriptions. In addition, the forthcoming OMG specification on BSBR Vocabulary will give birth to domains specific SIGs about finance, space, insurance, medical and so forth that will allow to import domain-specific vocabularies.

The Servent has the ability to automatically expose DBE services through a SOAP interface and to easily create adapters for other technical interfaces.

The Nervous System has to be multiproxy-capable; this means that it supports the distribution of platform-specific proxies like .NET or Java.

7 Structural Architecture

The main goal of the structural architecture, as described at the beginning of this document, is to support the realisation of the required functional features enabling the project for extra-functional features like reliability, security, logging, performance, long-running transactions and so forth. As the functional architecture has to collect and define the functional features, the structural architecture has to provide the same objectives for the extra-functional features.

The DBE has ambitious goals also from the point of view of the extra-functional features. Essentially the project has to be as reliable as the Internet and at the same time to be owned by the community of users themselves. The DBE, as an ecosystem, is constituted by its parts and relationships and there must be no single owner or controller. This approach in the DBE definition has a two-fold advantage:

- its infrastructure is reliable;
- there is no "big brother".

The infrastructure reliability (if properly implemented) allows the DBE not to depend on the failure of some nodes or network cables. The ecosystem will be to some extent able to support the partial failure of its constituent physical parts. As a matter of fact, the decision of adopting an Open Source approach, which is a technical architecture decision (Ref. Chapter "Technical Architecture"), goes in the same direction and allows SMEs to own both the software infrastructure code of the DBE and their own services.

Regarding the second point, the SMEs will be confident that the system governing laws of the ecosystem are not affected by the decisions of a single entity, company or committee. Each functional feature and QoS function is conceived to be open and not in the hand of a single entity.

In order to achieve these challenging features, the DBE needs to be at the cutting edge in pervasive distributed architectures. The first goal of the structural architecture is to realise all the storage systems, essentially the model repository, the semantic registry, the evolutionary environment and the service proxy registry, using distributed storage space techniques. Secondarily there is the need to provide a proper middleware subsystem with synchronous and asynchronous capabilities; we call this middleware the "DBE Nervous System". It is conceived to act as a glue between DBE business services and its structural elements.

Given the high decoupling requirements, the asynchronous mechanism will be based on the publish/subscribe schema. This approach allows services and components, that did not know of each other's existence beforehand, to exchange messages and receive events. One of the most important advantage of the p/s schema, beside avoiding hard-coding the dependencies, is that it avoids sending broadcast messages. For example if a user is interested in being notified of new services being published, it will ask the Nervous System to be notified whenever this kind of event happens. The Nervous System will keep a list of event listeners and will send notification messages each time the subscribed event happens[10].

This mechanism is a powerful way to enable feeding the evolutionary environment in a transparent way from the usage events happening concurrently in the Execution

10 This short explanation does not intend to be describe the entire publish/subscribe event system.

Environment. It is considered an essential feature to strongly decouple the dependencies between the ExE and the EvE in such a way that the first is not aware of the existence of the latter: the asynchronous bus of the Nervous System will help this mechanism to be realised. The goal of the technical architecture is to provide the best implementation by adopting a proper software framework.

The current state of the art in distributed computing in a highly decoupled environment, is Web Services. Beside the weakness in the business specification, as described in other sections of this document, there is also, in our perspective, an issue related to how actual services are reached and invoked. As in the legacy approaches to RPC (Remote Procedure Calls) the services execution points are treated as Internet endpoints (IP addresses) despite the fact that Internet IP addresses are expensive resources and are not available to the majority of the users. Most of the services make use of dynamic IPs and are not available 24/7, hence a better schema for reaching services is required. Moreover, the mobile technology and wireless systems are increasingly making use of dynamic addresses that change at every connection and at every cell switch.

The DBE Nervous System, as opposed to the Web Service approach to service invocation, makes use of a smart proxy. This pattern in remote call invocation is not new and can be found in CORBA, RMI, Jini and .NET. A smart proxy is a class, instantiated at the client site, that holds a reference to a remote object. It implements the service remote interface and typically forwards most of the calls on the interface remotely. However, a smart proxy is more useful than a plain delegator pattern (as is the case for Java RMI); it can change the behaviour of the remote interface. For example, it can locally cache the states of the remote object to avoid the network overhead on every method call or to forward the calls to a different end-point based on its availability or overhead.

Network topology

In Figure 22 below the structural topology of the DBE is shown. The figure shows the existence of three network layers:

- SF and modelling
- memory and repositories
- ExE and consumption phase.

In addition there is the asynchronous bus the provides usage data to the EvE, connecting the memories and the ExE. The single direction arrow pointing from the Nervous System to the Servent means that the ExE has never the onus of sending usage information or statistics indirectly to the various memory; this would jeopardize the performance of the entire system.

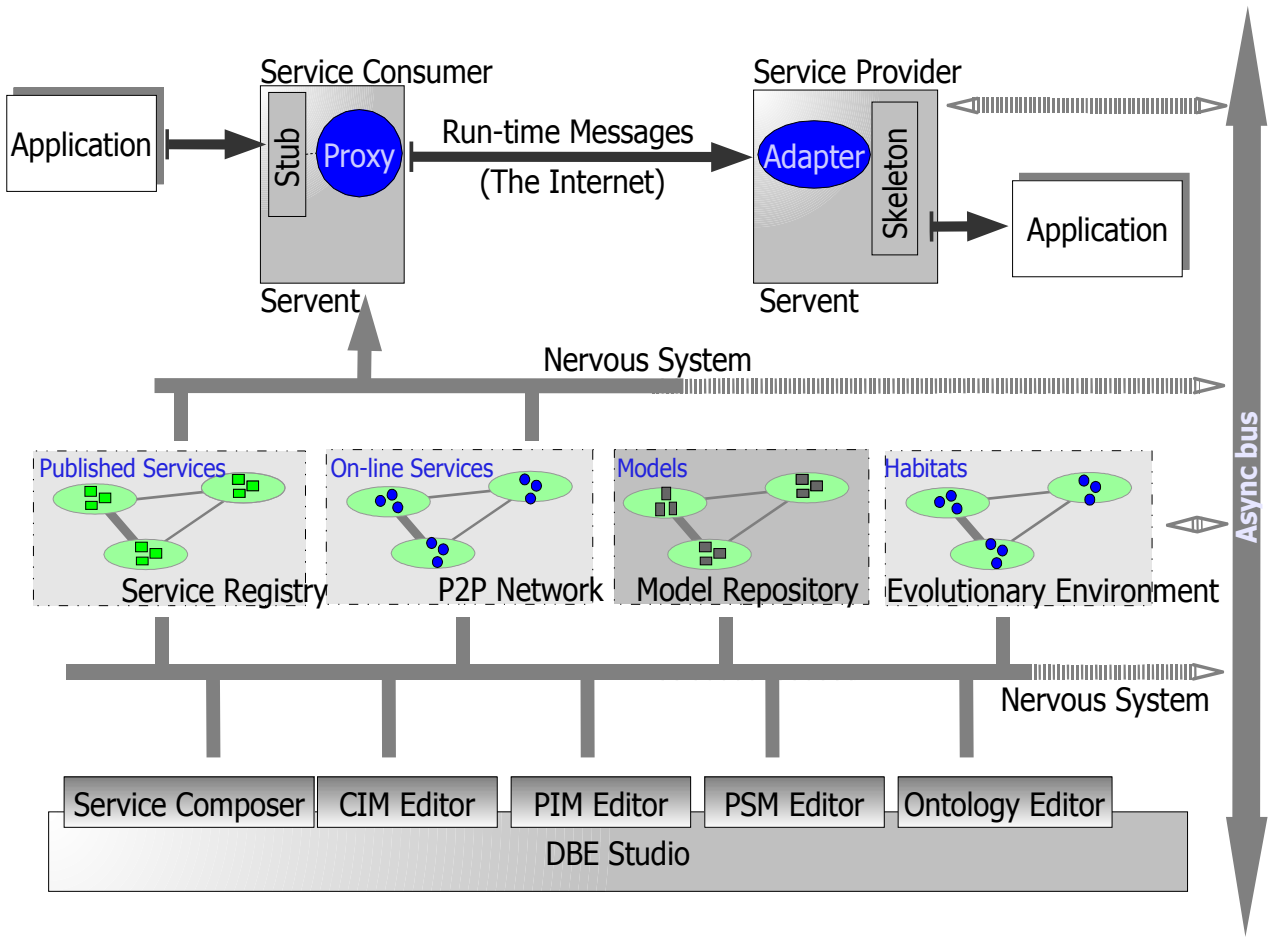


Figure 22 - Simplified Network Topology

Figure 23 - DBE Nodes mapping to DBE Environments shows the DBE network as a map of nodes: the figure represents an hypothetical instant in time. From this perspective the DBE is a collaborative network of nodes, each of which has one or more role.

In detail, a one-to-one logical relationship exists between an SME user and their EvE Habitat. However to manage the processing requirements, taking advantage of the distributed computing resources, an SME's Habitat will be instantiated where it is most efficient, or even across multiple SME Servents if it contains many evolving populations. This is shown on the Figure above, except for the latter scenario. This will render the distributed evolutionary optimisation more efficient in the presence of habitat clustering and close cooperation between SMEs.

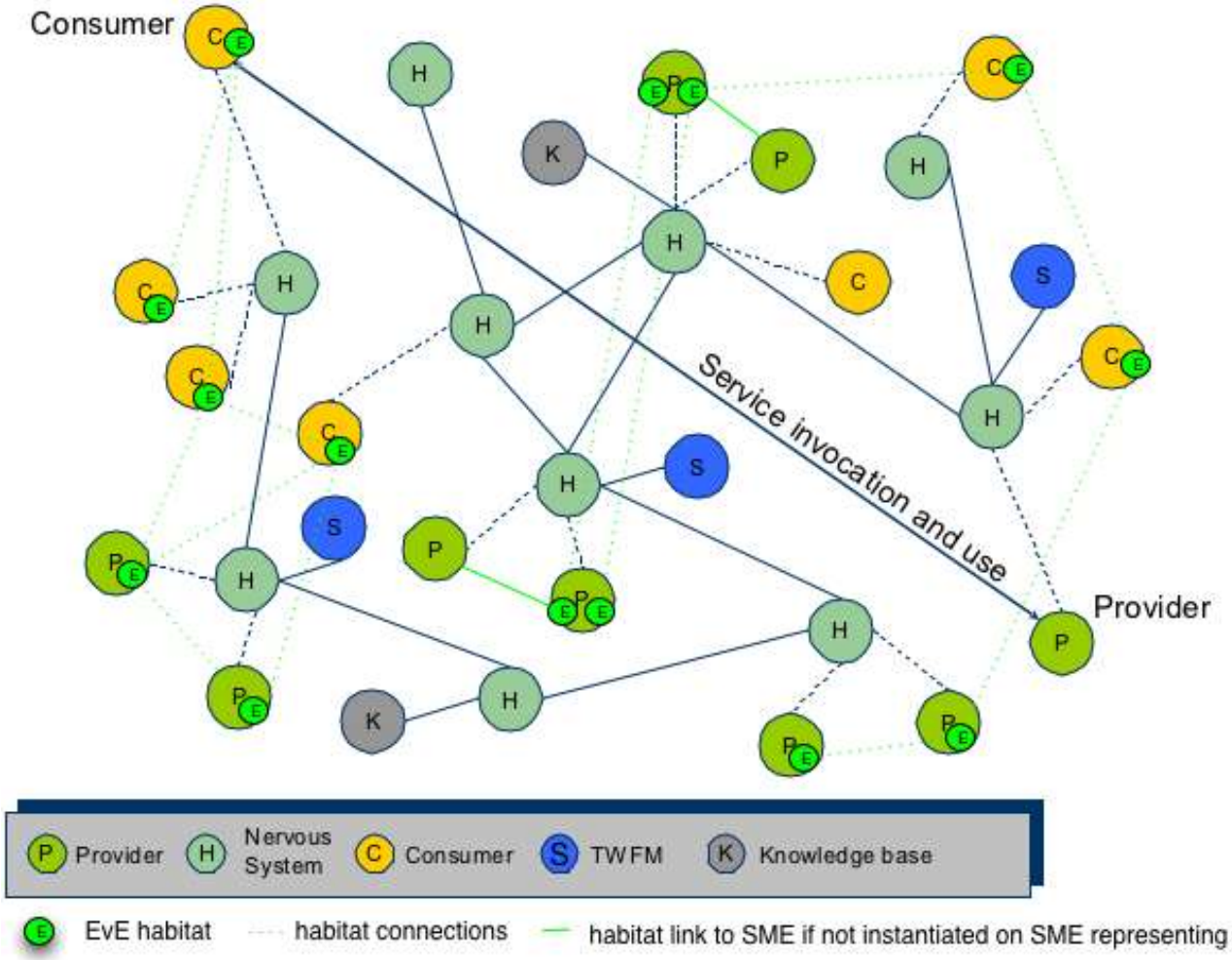


Figure 23 - DBE Nodes mapping to DBE Environments

8 MDA Testing approach

The DBE Service Factory intends to fulfil most of the needs of SMEs Software Developers who intend to deploy a service. The environment leverages the MDA approach and offers a facility for testing services. The SF can generate test cases based on BML and SDL models. This means that the SME Software developer can automatically obtain test data for feeding its service from the service specification. This can serve the testing purpose both at the business and the technical viewpoint.

Figure 24 - MDA based test generator below describes the approach in more detail. The DBE consortium, base on the actual BML and SDL metamodels, creates an application (MOF Test Generator "generator") that, provided as a SF plugs-in, can generate a test generator ("Test Generator"). This is turn receives service models as input for generating either test cases and unit tests.

The SF will contains a Test Generator for the BML and the SDL. Such an architecture facilitates the adaptation of the test generator to changes in metamodels.

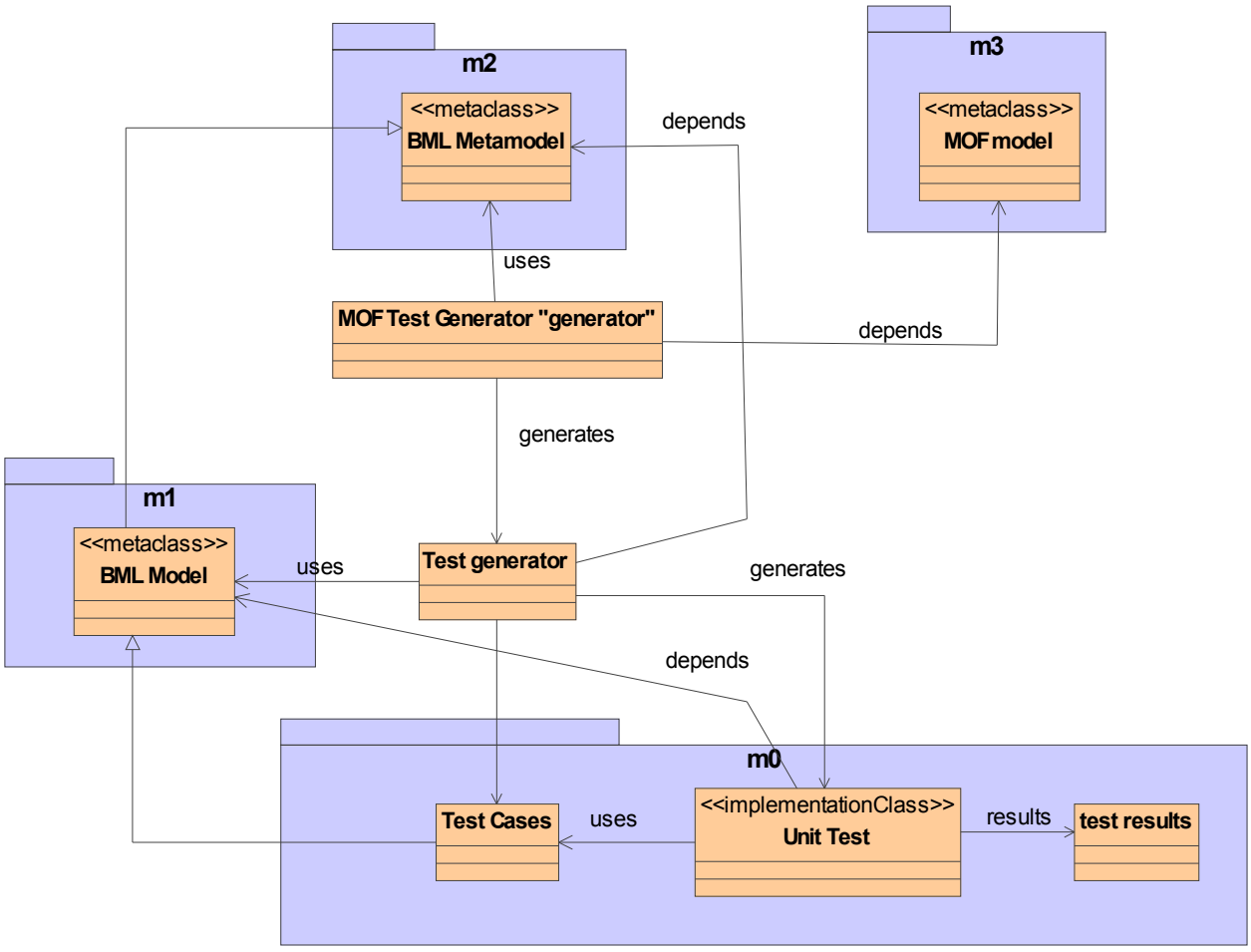


Figure 24 - MDA based test generator

9 Relevant similar efforts

9.1 Ebay

www.ebay.com

Classification

Ebay is the most famous Market Place in the world.

Functionality

EBay supplies services for:

- Trust and Safety
- Payments
- Member Reputation
- Bidding Tools
- Protection and Authentication
- Store

Standard

Web services

Interoperability

The eBay SDK allows to build applications interacting with eBay using Web Services. The libraries can be used from any .NET-based or COM-enabled language, including Visual Basic 6, Visual Basic .NET, C#, Perl, ASP .NET, PHP, and C++.

More information can be found in http://developer.ebay.com/DevZone/docs/SDK/

Ebay exposes Web Services using two different protocols XML/HTTPS and SOAP.

Technology

Business model

License OSS

Ebay is not open source; the eBay SDK it's only for Members who have to pay an annual fee to join the Developer Program.

9.2 Pax

www.vtt.fi/tte/proj/pax/

Classification

Automatised web service execution framework.

Functionality

Pax allows process-based web service composition.

Standard

Interoperability

The project aims at standardised descriptions for the processes so that software agents might be able to select, execute, monitor, and terminate processes

Technology

not enough information available

Business model

not enough information available

License OSS

not enough information available

9.3 Simile

http://simile.mit.edu/

Description

SIMILE means Semantic Interoperability of Metadata and Information in unLike Environments.

"SIMILE is a joint project conducted by the W3C, HP, MIT Libraries, and MIT CSAIL. SIMILE seeks to enhance inter-operability among digital assets, schema/ vocabularies / ontologies, metadata, and services." [From simile.mit.edu/]

Classification

Development process and tool.

Functionality

Simile is a set of tool:

- Longwell: a suite of web-based RDF browsers.
- Haystack: is called universal information client. "Haystack is designed to make it easy for developers to build powerful Semantic Web applications for end users, incorporating both code for processing RDF, ontologies/schemas, and user interface components. In conjunction with our source releases, we will release documentation on the structure of the system and its programmatic interfaces." [http://haystack.lcs.mit.edu/documentation.html]
- Gadget: a command line XML inspector. Gadget supply Functionality for data migration/transformation, data cleanup, data complexity evaluation and schema adherence understanding.

- RDFizers: is a set of tools that allow to transform existing data into an RDF representation. Really interesting the java2rdf that scans java bytecode and dumps a description of the dependencies and the package/archive containment.
- Welkin: is a graph-based RDF visualiser.

Standard

RDF, RDF Schema, DAML + OIL, OWL

Interoperability

Import/export different Ontologies stored using different schema.

Technology

All the Simile tools are built using Java.

Haystack is an Eclipse project (not delivered as plugin)

DSpace is used to store Ontologies.

Business model

License OSS

The project and the documentation are released under a BSD-style license. [http://www.opensource.org/licenses/bsd-license.php]

9.4 Satine

http://www.srdc.metu.edu.tr/webpage/projects/satine/

Description

The project is still on going but some interesting publications are available in the project site.

"The objective of the project is to develop a secure semantic-based interoperability framework for exploiting Web service platforms in conjunction with Peer-to-Peer networks in tourism industry".[http://www.srdc.metu.edu.tr/webpage/projects/satine/]

Classification

Semantic P2P Framework

Functionality

The SATINE project is a semantic-based interoperability framework for the tourism industry. The framework will provide tools and mechanisms for publishing, discovering and invoking Web services through their semantics in peer-to-peer networks. All the services have to be described with the same ontology.

Satine supply security using XML Signature, XML Encryption and WS-Security.

Standard

RDF, WSDL, UDDI, SOAP, JXTA.

Interoperability

A semantic WS mapping is supplied to map services using different ontologies.

Technology

Satin use a P2P Technology (with SuperPeer topology) based on the JXTA Framework. The services will be deployed as WebServices defined using WSDL and accessed using SOAP.

Business model

License OSS

9.5 OBELIX

http://obelix.e3value.com

Description

The project developed the OBELIX ontology tool suite for smart collaborative e-business.

The suite is composed by a generic ontology server, an ontology library for smart collaborative e-business, and several ontology-based application tools.

Classification

Ontology tool suite.

Functionality

The Obelix project has developed a lot of ontology tools:

OntoEdit allows to edit ontologies, provides aligning and Merging facilities.

OntoMap is a graphical ontologies mapping tool. It allows to graphically draw mappings between concept, attribute or relation belonging to different ontologies.

OntoVersion is a versioning tools for ontologies.

OntoServer is the ontology server is the run time system which administrates the knowledge base containing the ontologies.

OntoBroker is the reasoning mechanism of OntoServer.

Standard

The ontology is presented using the RDFS-W3C standard.

Interoperability

Currently OntoServer reads F-Logic , OXML, a Datalog/Prolog like syntax and RDF(S).

The OntoServer expose a Web Services interface and provide access for Jscript,

Python / Zope, ASP,Visual Basic, JSP, PHP

Technology

Obelix project developed a series of Java plug-in and a Java OntoServer.

Ontobroker may be configured as a distributed system where different ontobroker servers collaborate on different computers.

Business model

license OSS

NO – Commercial info http://www.ontoprise.de/products/

9.6 Universal Design of Digital City

http://www.digitalcity.jst.go.jp/index.html.en

Description

The project start from the consideration that people are starting to use the Internet, not only in business but also in their daily life. The objective of the project is to create digital cities as an infrastructure used and participated by all people, including the handicapped and the aged. This project might be used to improve the e-government and the e-democracy.

Classification

Virtual city

Functionality

Build high quality 3-D models of a town.

Define language for describing interaction between agents and users.

3D chat, multi-user training and visual simulations.

Standard

Interoperability

The system doesn't aim at integration to outer applications.

Technology

Digital Cities have used technologies such as geological information systems, virtual space and mobile computing. The project use the perceptional information infrastructure (PII) and the software called social agents. The main technological effort aim at image rendering and virtual space representation.

Business model

License OSS

JST LICENSE (http://www.digitalcity.jst.go.jp/~satoshi/TownDigitizing/)

10 DBE Components

Term	Description
Authoring tool	(formerly BML Authoring tool) is used to assign values and data in general to a BML model
BML Editor	A tool to model the concerns of the business. The modelling task might be graphical or rule base; it is not a matter in the context of this document. What is important thought is that the output model is an instance of a BML metamodel.
BML Wizard	Part of the BML Editor that is meant to ease the realisation of BML models by inspecting the catalogue of models and suggesting the more appropriate
Business Service	The actual service supported by the DBE that is not either structural or basic
DBE Composer	A tool for creating a workflow using pre-existing DBE Services
DBE Desktop	A tool (or more precisely a feature of the DBE Composer) for maintaining a list of services that are used by SME Consumers. In the DBE Desktop, as opposed to the Composer, services are not chained or formally connected together, they are simply aggregated. The DBE Desktop resemble a classical URL Bookmark Manager.
DBE Portal	The web site of the DBE where users can start understating what it is. Beside marketing documentation it provides a location from which to download the components (super nodes, development environments, Servents, ...). It also provides a window to the runtime: repository browsing, service browsing, active FADA nodes, network topology... The DBE Portal is expected to provide some DBE components in ASP mode like the DBE Desktop and the Service Browser (possibly also the executor).
DBE Studio	The GUI based tool that organise the different tools used in the development environment. It is the implemented development environment.
ExE	Ref. Service Execution Environment
FADA	Federated Advanced Directory Architecture, an Open Source implementation of a P2P network that has it origins in the SUN' Jini specification.

Term	Description
Form Designer	A tool that ease the drawing/definition of a GUI form. The resulting GUI form is to be used for executing/consuming a service.
Interaction Form	The resulting output of the Form Designer. The resulting GUI form is to be used for executing/consuming a service.
KB	Ref. Knowledge Base
Knowledge Base	The Knowledge Base represent the DBE pervasive storage area that contains: models, metamodels, Service Manifests, non-structural data (like a BPEL), ontologies and anything else that will ever find reasonable to store. The Knowledge Base is the Service Factory Environment memory storage area. The Execution Environment memory space is essentially the Semantic Registry.
Model Repository	The part of the Knowledge Base that stores the Models used in the DBE
Recommender	The process that given a query request, is able to inspect the Model Repository and providing a ranked list of results. The recommendation process can retrieve models or Service Manifests.
Repository	Ref. Model Repository
SDL Editor	An editor to create SDL models of a service. It will model concepts like: service name, operations, parameters, exceptions and so forth.
Semantic Service Language	Semantic Service Language, the metamodel for describing service semantic
Servent	The DBE Application Server: Ser(er)(cli)ent. It server the objective of hosting both a service proxy (the front-end) and the adapter (back-end mediator to the service)

Term	Description
Service DNA	It is the pair: SDL and BML model. It represent the conceptual definition of a service when it is not related to any real service. By "real" we mean a service that exist in the real world: for example the supplier of a real service has a VAT number, a snail mail address or an IP address. The binding defines the tangible relation with a real service. The Service DNA is an element of reuse across services. It is the conceptual definition of a service when not related to any real service; it represents and element of reuse (aka Service Definition).
Service Execution Environment	It is where services live, where they are registered, deployed, searched, retrieved and consumed. This parallel word is sometimes referred to as the "run-time of the DBE".
Service Factory Environment	is devoted to service definition and development. Users of the DBE will utilise this environment to describe themselves, their services and to generate software artefacts for successive implementation, integration and use.
Service Manifest	The container that entirely describe a service, the difference with the Service DNA is the represent of service information (M0 in MDA terms)
SFE	Ref. Service Factory Environment
SM	Ref. Service Manifest
SME Consumer	It represent an SME aware user that search and consumers a service in the DBE
SME Provider	It represent an SME aware user that offers a service in the DBE
SME Software Developer	It represent an SME aware user that creates the software artefacts in order for a service to be published or consumed in the DBE
SME User	It represent a generic SME aware user that interacts as a service provider or service consumer
SSD	Semantic Service Description: the Model build using the SSL
SSL	Ref. Semantic Service Language
Structural Service	Support service like, Repository, Accounting, Security, Distribution, transactions...
Transactional Work Flow Manager	An application able to execute a service workflow.

Term	Description
TWFM	Ref. Transactional Work Flow Manager

11 Glossary

Term	Description
ASP	Application Service Provider
b&b	Bed & Breakfast
B2B	Business to Business
B2C	Business to Consumer
Back End	In client/server distributed computing it is a generic term that refers to the runtime part of an information system that runs remotely with respect to the client hardware
Back Office	Department that has no or less contacts with the customers. From the wikipedia: "A back office is a part of most corporations where tasks dedicated to running the company itself take place."
Basic Service	Basic Services are DBE service that pertains to the following categories: payments, information carriers. The list could increase further on.
BE	Ref. Back End
BML	Business Modelling Languages
BO	Ref. Back Office
CIM	Computational Independent Model
CWM	Common Warehouse Metamodel
DBE	Digital Business Ecosystem
DNA	deoxyribonucleic acid
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EDOC	Enterprise Distributed Object Component
FE	Ref. Front End
Front End	In client/server distributed computing it is a generic term that refers to the runtime part of an information system that runs locally with respect to the client hardware
GUI	A Graphical User Interface, it is a kind of UI
M0	MDA layer that references to M1 based data ('Mike Phone has invoice #1221')
M1	MDA layer that references to M2 based models ('customers may have invoices')

Term	Description
M2	MDA layer that references to MOF models (e.g. UML, CWM, EDOC, BML, SDL...)
M3	MDA layer that references to MOF language
MDA	Model Driven Architecture
MOF	Meta Object Facility
OMG	Object Management Group (www.omg.org)
P2P	Peer to peer
PIM	Platform Independent Model
PKI	Public Key Infrastructure
Proxy	Executable Java object that can be distributed over the Internet. It is usually a mediator to the actual remote service that provides the required functionality.
PSM	Platform Specific Model
RUP	IBM© Rational Unified Process®, or RUP®, http://www-306.ibm.com/software/awdtools/rup/
SDL	Service Definition Language: a language for the definition of a Platform Independent Model (PIM) of the service interface
Service Proxy	Ref. Proxy
SM	Ref. Service Manifest
Smart Proxy	Ref. Proxy
Super node	In P2P technology it represents a node of the network (a remote server). In the DBE there are several kind of super nodes: KB, FADA, DSS.
UDL	Universal Design Language
UI	User Interface
UML	Unified modelling Language
User Interface	Graphical User Interface
UUID	Universal Unique Identifier
VAS	Value Added Service: a service that is the aggregation of other services.
X509	It is a standard that makes it possible to identify someone or something on the Internet.
XMI	XML metadata Interchange
XML	Extensible Markup Language

Term	Description
XP	Extreme Programming http://www.extremeprogramming.org/
XUL	XML User Interface Language
ZIFA	Zachman Institute for Framework Advancement, http://www.zifa.com

12 Bibliography

[BML metamodel]: Angelo Corallo ed others, "BML Metamodel v0.1", 2004
[BML metamodel2]: Angelo Corallo ed others, "Work Package 15 DBE Business Modelling Language v2.0.1", 2004
[BMLWS2003]: P. Dini, T. Kuusisto, A. Corallo, P.Ferronato, N. Rathbone, "Toward a semantically rich BML for the automatic composition of WS", 2003
[CANTU00-1] Cantu-Paz, E. and Goldberg, D.E. (2000). Parallel genetic algorithms:theory and practice. Computer Methods in Applied Mechanics and Engineering. New York: Elsevier.
[CANTU00-2] Erick Cantú-Paz (2000) Efficient and Accurate Parallel Genetic Algorithms Kluwer Academic Publishers
[CANTU99-1] Erick Cantu-Paz and David E. Goldberg (1999), On the Scalability of Parallel Genetic Algorithms, In Evolutionary ComputationVolume 7, Issue 4
[CANTU99-2]
[COHOON] Cohoon J P, Hegde S U, Martin W N and Richards D S (1987), Punctuated Equilibria: A Parallel Genetic Algorithm., Proc. 2nd Int. Conf on Genetic Algorithms (Pittsburgh, PA, 1987), ed J Grefenstette (Hillsdale, NJ: Erlbaum)
[COOPER]: Alan Cooper, "The Inmates are Running the Asylum", 2003
[DBE]: F.Nachira, E. Chiozza, H.Ihonen, M.Manzoni, F. Cunningham, "TOWARDS a NETWORK OF DIGITAL BUSINESS ECOSYSTEMS", 2002
[EvE]: "Evolutionary Environment Architecture Requirements", Gerard Briscoe, Intelligent Systems and Network Group Department of Electrical and Electronic Engineering Imperial College London, May 2005.
[FER04]: Pierfranco Ferronato, "DBE Core Scoping Architecture", 2004
[Fetish]: http://www.t-6.it/fetish, http://ica.cordis.lu/search/index.cfm?fuseaction=proj.simpledocument&PJ_RCN=5413515&CFID=967518&CFTOKEN=46427278
[FRK]: D.S.Frankel, "Model-Driven Software Development", MDA Journal, www.bptrends.com, 2004
[GORDON] S. Gordon and D. Whitley (1993), Serial and Parallel Genetic Algorithms as Function Optimizers, International Conference on Genetic Algorithms. S. Forrest, ed. Morgan Kaufmann.
[HS2002]: Peter Herzum, Oliver Sims, "Business Component Factory", 2002
[INES]: Ines Alves de Queiroz, "SME Use case – Leisure services", draft version, 2004
[Keller]: Wolfgang Keller, "The pitfalls of Meta-Systems and Business Rules", www.objectarchitects.de, 2004
[MDA]: OMG, Architecture Board ORMSC, "Model Driven Architecture (MDA)", ormsc/2001-07-01, July 9, 2001
[MDAF]: D.S.Frankel, "Model Driven Architecture Applying MDA to Enterprise Computing", Addison Wesley, 2003
[MDAGuide]: OMG, "MDA Guide Version 1.0.1", http://www.omg.org/docs/omg/03-06-01.pdf, 2003
[MDAK]: Anneke Kleppe, Jos Warmer, Wim Bast, "MDA Explained", Addison-Wesley, 21st April 2003
[MITCHELL] Melanie Mitchell (1998) An Introduction to Genetic Algorithms, Bradfrord Books.
[MOF1]: OMG, "Meta Object Facility (MOF) Specification", v1.3 March 2000, 00-04-03.pdf

[OPAL]: A.Formica, M.Missikoff; Ontology Validation in OPAL, Proc. Of the Int. Conference on Web Services (ICWS), L.J.Zhang (Ed.), pp.198-202, LasVegas, Nevada, Jun 2003.
[OPAL2]: Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001 table of contents Ogunquit, 2001, ISBN:1-58113-377-4
[QML]: "WP17: Composer, D17.1: Recommender", DBE Deliverable, Technical University of Create, 15/02/2005
[RUP]: "Philippe Kruchten", The Rational Unified Process: An Introduction", Addison-Wesley Pub Co, ISBN: 0321197704, December 19-2003.
[SCHLEUTER]Gorges-Schleuter M ed H-P Schwefel and R Männer (1990), Explicit Parallelism of Genetic Algorithms Through Population Structures, Parallel Problem Solving from Nature (Berlin: Springer) Lecture Notes in Computer Science
[SIMS94]: Oliver Sims, "Business Objects", McGraw-Hill, ISBN: 0077079574, October 1, 1994
[SOL04]: "DRAFT_SMSoftwareModel", Soluta.net, G.Montanari, V.Trentin, 2005/01/02
[STU04]: "Service_Manifest_Conceptual_Model_V3", Salzburg Technical University, C. Masuch / 2004-05-03, internal DBE document
[TA]: "SIXTH FRAMEWORK PROGRAMME PRIORITY 2.3.1.9. Networked Business and Governments, Annex I - Description of Work", IST - 2002 – 507953, 29.10.2003
[TIMBML]: Tim Romberg, "Very Simple BML", paper, 2004
[TIMH]: Tim Romberg, "Hotel – Corporate client scenario", paper, 2004
[WHITLEY] D. Whitley, S. Rana and R. B. Heckendorn (1997), Island Model Genetic Algorithms and Linearly Separable Problems, Proceedings of the AISB Workshop on Evolutionary Computation.
[ZIFA87]: "A Framework for Information Systems Architecture." John A. Zachman. IBM Systems Journal, vol. 26, no. 3, 1987. IBM Publication G321-5298. 914-945-3836 or 914-945-2018 fax.
[XP]: Kent Beck, Cynthia Andres, "Extreme Programming Explained", Addison-Wesley Professional; 2 edition, ISBN: 0321278658, November 16-2004.
Cantu-Paz, E. and Goldberg, D.E. (1999). On the scalability of parallel genetic algorithms. Evolutionary Computation. 7(4).
[WP 21] DBE Architecture Requirements

- end of document -