D.B.E.

Digital Business Ecosystem

Contract n° 507953

WP16. Service Description Language

D16.1 Service description models and language definition

Part 2 - Service Manifest Conceptual and Software Models

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

Contract Number: 507953

Project Acronym: DBE
Title: Digital Business Ecosystem

Deliverable N°: D16.1
Due date: 30th April 2005
Delivery Date: 14th October 2005

Short Description:
This document describes the Service Manifest Conceptual Model and the Service Manifest Software Model.

Partners owning: Soluta.net
Partners contributed:STU - chapter 5-6
Made available to: DBE Consortium

Versioning		
Version	Date	Name, organization
00.02	2004/07/25	Pierfranco Ferronato, Valentino Trentin – Soluta.net
00.03	2004/12/20	Valentino Trentin, Giulio Montanari – Soluta.net
00.04	2005/06/22	Andy Neagu, Giulio Montanari – Soluta.net
00.05	2005/08/31	Giulio Montanari, Valentino Trentin – Soluta.net
01.00	2005/08/14	Giulio Montanari, Valentino Trentin – Soluta.net Claudius Masuch, Thomas Kurz, Giulio Marcon - STU

Quality check
1st Internal Reviewer: Miguel Vidal – SUN Microsystems
2nd Internal Reviewer: Angelo Corallo - ISUFI

1Table of Contents

2List of Figures

3Executive Summary

The Service Manifest (we will refer it as SM) is the complete representation of a DBE Service. All the information required to describe, retrieve and execute a service are contained in the SM.
Because the DBE uses the MDA[1] approach the information contained by the SM is structured using models which define the information from different points of view[2].
The SM contains information about service from two points of view, a business[3] and a technical[4] one. From the business point of view the information is modelled using BML[5], which describes the business model of a supplier and the service itself from this particular perspective. From the technical point of view the information is modelled using SDL[6] which describes the technical interface of the service at a high level of abstraction independent of any technical implementation. This representation of the service addresses the computational structure of the service. The information described using SDL is used to generate the Java code[7], which implements the proxy of the service.
The models (non computational and platform independent) represent the conceptual definition of the SM and they are called Service DNA (we will refer it as SDNA).
The services represented by a SM might be a Simple Service or a Service Chain. A service chain is a composition of services that is expressed using a language called BPEL[8]. The SM doesn't contain any platform specific information being like this completely platform independent.
Besides the SDNA, the SM stores information about the service supplier and the actual service itself. In other words, besides its CIM and PIM, the SM stores instance and even quality of service data[9]. The SDNA can be seen as the conceptual model of the service while the SM is an instance of that service which is directly related to a specific supplier and a specific implementation.
All the models contained in the SM contain links to ontologies because these references allow the discovering, reasoning and the comparison between different services.
Syntactically, the SM is represented by an XML[10] file, and the models contained are stored using an MOF[11]-XMI[12] representation.

1 Model Driven Architecture [MDA].
2 Usually three [MDA].
3 Computational Independent Model (CIM) [MDA].
4 Platform Independent Model (PIM) [MDA].
5 Business Modelling Language
6 Service Description Language
7 Java is the chosen technological platform, for the moment.
8 Business Process Execution Language
9 See [DBECoreArch].
10 Extensible Markup Language [XML].
11 Meta-Object Facility [MOF].
12 XML Metadata Interchange [XMI].

4Foreword

This paper addresses some of the principles that we believe are strategically important for the development of the Service Manifest, specifically regarding its structure. This is not a primer or a tutorial about SM or XML, we assume that all the partners are proficient with XML and XMI. This document is a living document for focusing concepts, for identifying features and approaches; it is based on a "JAD[13] sessions", a continuous brainstorming.
This paper had been rewritten several times until now because the requirements of the DBE component had been changed to address the features of other DBE components. Many suggestions made by this document were considered useful and developed in other documents.
Because some of the documents involved in the definition of the SM weren't published on time it wasn't always possible to bring up to date the present document. After the first 18 months almost all the documents will be completed and then it will be necessary to be done a process of review and integration.
It must be clearly understood that this document is not in its final version. There are still many concepts that have to be formally defined by the specific tasks before they can be included into the present document. The SM will change accordingly to new emerging ideas and to new researches that will be done inside the task itself. Some sections in this document are marked with TBD (To Be Developed) to indicate the areas that will be further developed.
This document contain an introduction to the concept of Service Manifest (Conceptual Model) and how it will be handled as a CIM. From the PIM point of view of the SM, there were considered principally the implementation details even if it was tried to address the general features of the SM.

13 Joint Application Development

5Service Manifest Conceptual Model

Within the DBE, each service is described by a so-called Service Manifest (SM). The SM is the single point of information for a given DBE-service. Figure 1 presents the model and parts of the SM. As can be seen in Figure 1, the SM comprises the three domains of the DBE, business, computing and science, resp. evolution.

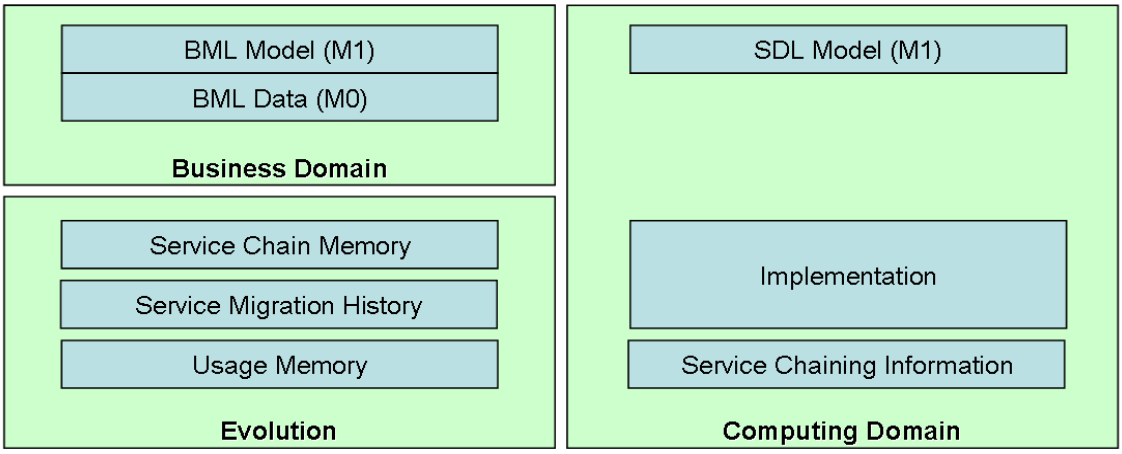


Figure 1- Service Manifest Conceptual Model

5.1Business Domain

The business domain part contains the BML model of the service. At Service Manifest level, a service as well as an organisation within the DBE are modelled and described using the same structure. But, as can be derived from the BML parts, the stored BML model itself contains this differentiation. As a result both service and organisation can be stored using the same structure. Moreover, extensions or modification of the BML Meta model do not have an impact on the model of the SM. Beside the BML model of a service, the SM also stores the appropriate BML data (see Fig. 2).

5.2Computing Domain

The contribution of the computing domain focuses on two main points. First, an appropriate SDL model for the BML model has to be defined. This model is stored within the SM. The SDL model defines the interface and therefore data types which are available for this given service.
Second, the SM can also store implementation parts of the service. In the easiest case, the SM is referenced by a proxy that points to the actual back office implementation. But, by design, the SM might also contain the implementation of the service. This might cover implementation parts like User-Interface factories, or even complete implementations that might come travel the SM.

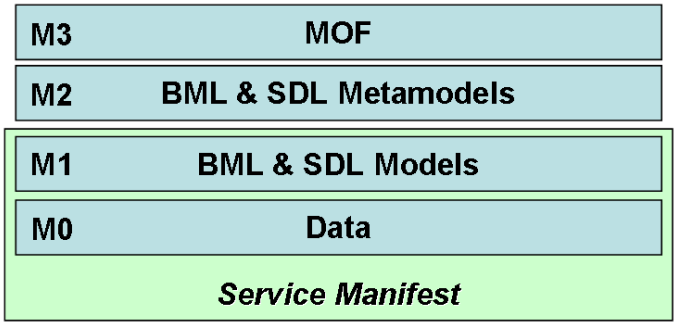


Figure 2- Service Manifest MOF Model

The work flow that drives service manifest chains is formulated using BPEL. As it is required for Service Manifest Chains, it is discussed later. MDA plays a vital role in the DBE and the DBE languages. Therefore, the languages parts (BML and SDL) of the Service Manifest are MOF-compliant. To be more accurate, the SM covers M1 and M0 level of BML, and M1 level of SDL. Figure 2 illustrates this.
The SM contains the actual BML and SDL model. There Service Manifest M0 level therefore comprises the BML data like prices or names.

5.3Evolution

The evolutionary part of the SM is intended to support the optimisation of service chains (see *Service Chains*) and to provide feedback capabilities for the service provider. This feedback should help to adapt the service to new or changed requirements of the user and improve the quality of the service. The evolutionary part of the SM covers three main topics: Service Chain Memory, Service Migration History, and Usage Memory. The Service Chain Memory data-structures store information about the service chains in which this service has been used. This comprises a list of chained services and the fitness that these service chains had.
The Service Migration History structure is to store information about the habitats, i.e. SMEs where this service has already been used, and information about the success. This information should allow other services to be approved to the current user. In particular, services that have been used within the same habitats and therefore could be of interest for the current habitats owner. The Usage Memory holds counters for the usage of a service. These counters include: number of offers to a SME, number of actually being used, service chain integration, and the usage of the service structure (BML M1) for the set-up of new similar services.

5.4Service Chains

A key concept of the DBE is to allow the composition and handling of service chains. A service chain is a DBE service that is composed of other DBE services. From a users point of view, there is no difference between an atomic (not-composed) DBE service and a service chain. The representation of a service chain again is a Service Manifest (SM). Hence, the user must neither take care about data transfer or transaction handling between the affected atomic services nor take care of the rules and conditions being subjected. This data transfer and transaction handling information, also referred as work flow, has to be stored in the belting Service Manifests Service Chain binding information (see Fig 1). This work flow drives the execution of a service chain. Figure 3 illustrates

this. Service D (Fig. 3) represents the DBE service chain that consists of service A, B, and C. These three services are executed in succession, whereas output from service A acts as input to service B, and output from service B acts as input for service C. If, apart

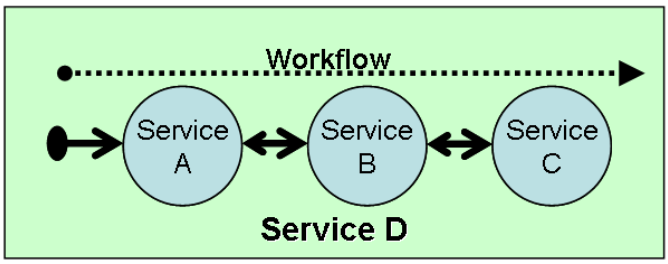


Figure 3- Service Chain

from this data flow, there is no other implication between these services, no additional information is needed. Each service is processed one after the other, using its predecessors output as input. For more complex data flow, additional information might be stored into Service Manifest D (for example, transactional processing, conditions to execute all services etc.). For the first implementation just sequential service chain concatenations are considered.

5.5Adaptation of Service Chains

By now, it is not possible in the DBE to actually adapt an atomic service to changing requirements. The usual design for a DBE service is that the Service Manifest (SM) proxy reference (see Fig.1) points to back-office implementation of the actual service. So in general, it is not possible for an end-user of such a service to change the service itself as there is no possibility to change the back-office implementation[14]. This scenario changes when thinking of service chains: here, an adaptation of the service chain can be achieved by changing one or more components of that service chain. In the same way as there exists more than one company for e.g. car rental, there will be more than one provider for a given service (in this case, a “rent-a-car” service). So the adaptation of a service chain means to find better fitting components that then lead to the designated adaptation. This adaptation is also referred as optimising a service chain, as the adaptation of the chain should bring a more optimal solution to the user. Figure 4 shows an example for this. This example could represent services that are used by a travel agency that offers a holiday bundle consisting of flight, hotel reservation, and reservation of a hired car. Any of these services may be provided by more than one supplier. For the matter of simplification, this example focuses on adapting just one component of the chain. In this case, a better solution by changing the car-rental service is aspired.

14This might be not entirely true, as a customer that uses a software product might recommend
changes of this product from the manufacturer. Anyway, these recommendations usually do not result
in an immediate change.

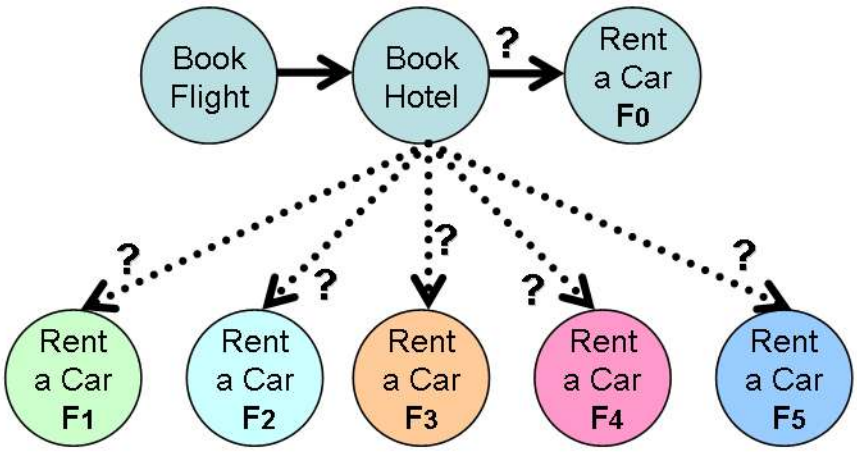


Figure 4- Service Chain Adaptation Example

Service chain adaptation depends on the relative "fitness" of the service chain in proportion to the user's preferences. In this example, the service chain might be fitter for the travel agency if the overall price is at a minimum. To achieve such a goal, the DBE might automatically or on demand change the Car Rental A with any other Car Rental and re-calculates the fitness - here, the price - of this service chain. This adaptation can be enhanced using more sophisticated mechanisms, and by expanding the service exchange to flight and hotel service.

6 Adaptation and Extension Capabilities

This chapter proposes and describes methodologies and features that are out of the current working scope of the project, but might be of interest for further developments of the Service Model in the DBE.

6.1Adaptive Service Model

The adaptation of a service is based on the concepts and ideas of the Model Driven Architecture (MDA) and the Adaptive Object-Model (AOM) Architecture[AOM]. The initial point for the adaptive service approach, which will be named Adaptive Service Model (ASM), is the Service Manifest Conceptual Model. Therefore, it uses the languages and data structures that come with a Service Manifest (see Fig. 1). The main idea that AOM brings is that models and meta models are instantiated in an execution environment. In the DBE, the BML and SDL are the languages that actually define the behaviour (BML) and interfaces (SDL) of a service. The BML meta model is capable of describing business entities, relationship, rules and behaviours by its process, organisation, locations, events, and motivation packages. According to the AOM approach, these business ontology ingredients are used to define the service-meta model. Even more, each business has sensors and effectors, meaning that a company is influenced by its environment and influences its environment. One a less abstract level, a business organisation or service in the DBE must be able to exchange information. In the SMCM, this information interface is defined by the SDL. According to MDA, M1 level of a service therefore is as used for run-time as well as design-time. So M1 has platform specific models (run-time), platform independent models (SDL Service Model) and the computational independent model of BML. Figure 5 illustrates this.

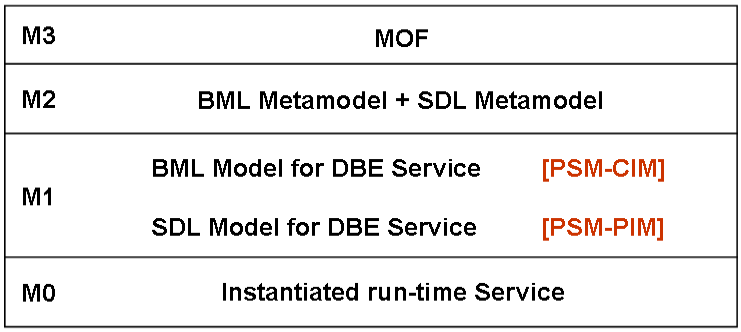
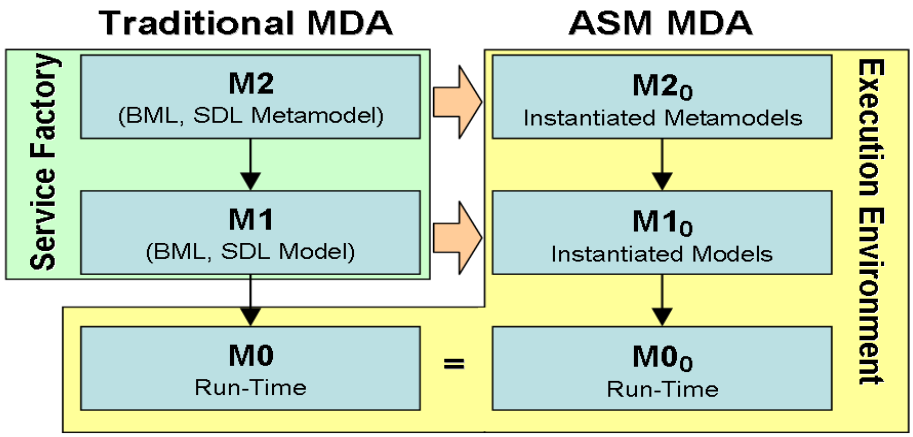


Figure 5- DBE Environments with ASM

Level M3 is MOF. At level M2, we have Meta models of BML and SDL. These meta models serve as the basis for the AOM – Meta modelling concept. Level M1, as instances of M2, is the instantiated service model and interface model of a service. As BML is computing independent, this is the Computation Independent Model (CIM) of our service. SDL, as it actually is the model for a computing interface, is the Platform Independent Model (PIM) of our service. In the Adaptive Service Model both models are also Platform Specific Models (PSMs) as they are actually run-time instances of M2, which means Platform Specific Models (PSMs), as run-time always means a specific platform. So adapting a service actually means modifying the BML model, which then

results in adapting the business behaviour of a service. Level M0, as always, is the implemented, instantiated, run-time service, that is used by a service. So the mix of MDA and AOM approaches actually means that we have two MDA views on a service: the “traditional” static MDA view of , and the AOM based run-time view if a service. Figure 6 shows both views and the transformation between it.



For the DBE the introduction Figure 6: DBE Environments with ASM of the Adaptive Service Model (ASM) means that the DBE Service Factory environment moves into the DBE Execution Environment. This combination means that the BML and SDL Models of a service are not just used to describe a service but actually to instantiate it, therefore merging the approaches of AOM and MDA to build a well-documented, highly maintainable, and adaptable service environment.

7Service Manifest Features

It's not easy to investigate the service manifest features since the architecture of the DBE languages is not 100% completed. At this time it is possible to address most but not all of the features; hence the SM will be improved over time during the life cycle of the project itself.

7.1Functional Features

Feature name	*Description*
Storing SDNA	The SM has to store the SDNA that in turn contains both CIM and PIM models.
Storing CIM Model	The SM is a composition of CIM Models of the service.
Storing PIM Model	The SM is a composition of PIM models; SDL will be used.
Storing service information (actual data)	The SM also contains service specific data like firm location, name, availability and so forth.
Link to the service proxy	The SM has to support the retrieval of proxies from the Nervous System (FADA[15]). Storing a link is an abstract concept, which means that the SM has to store some data in order to allow external processes to get the related proxies.
Link to Fitness data and Quality of Service	As for for the “Proxy link” above, the link to the Fitness Data and Quality of Service is an abstract concept, anyway the association between SM and Fitness data and Quality of Service is a composition.
Importing/Exporting models in XMI format	The SM has to enable the models import/export in XMI format.
Unique identification	Each SM has a unique identification key.

7.2Extra Functional Features

Feature name	*Description*
XML format	The SM is a XML document, not an XMI1.2/MOF1.4.
Tamper proof	The SM must be modifiable only by the creator/owner. Any attempts to change it from unauthorized users must be blocked.
Traceable	Any modification to the SM must be traced and the author identified.

15 See http://sourceforge.net/projects/fada/.

8CIM and PIM Models of the service

The MDA approach, used by DBE, states that should exist more than one model for the same system or subsystem that should be described. The number of points of view from which any system should be described were established at three. Then, there has to be a Computational Independent Model (or CIM), a Platform Independent Model (or PIM), and a Platform Specific Model (or PSM). The CIM is a business representation of the described system. The PIM describes the system in a computation able manner, but without any commitment to any existing technological platform. Finally, the PSM represents the executable form of the system for a chosen platform. Usually, there are more PSMs for the same system, each other running on a different technological platform.
Following the MDA approach the SM is described from different points of view, any of these descriptions being written using specific languages. These languages have to be different because they address different category of people which approaches the system from different points of view according with their field of expertise.
The business representation of the SM, or CIM, is written using BML and SSL[16], while the technical interface of the service, or PIM, is written using SDL. The PSM representation of the SM is written in Java and the code is generated automatically from the PSM representation of the service.
This document doesn't aim at discussing these languages, refer to [DBECoreArch] for an overview over the CIM and PIM concepts, to [BMLMM] for information about BML, to [KBDI] for information about SSL, and to [SDLDef] for further information about SDL.
Because it was decided that the SM should have some degree of autonomy, any SM contains both its CIM and PIM. That means that the business model written in BML and SSL and the technical interface of the service written in SDL are both included in the final definition of a service.
All the models representing the Conceptual Model of a service are stored in the KB Model Repository (it is a part of the KB) for further reuse. An exception is made by the PSM representation of the SM[17] which is not stored in the Model Repository, because it will be generated automatically by the Service Factory. When a DBE user wishes to deploy a new service it is strongly recommended to reuse an existing model either for the business models and for the technical ones. The reuse of the models promotes the emergence of de-facto standard services. Refer to chapter 9.2 - Evolutionary aspect of SDNA and “standard de facto” for further discussion about the reuse ad the standard services.
In brief, the sequence of operations that should be done to create a service is: first it should be created the SDNA of the service by creating the business model of the service using BML and its technical interface using SDL, next the service should be “instantiated” by setting the particular data which will define the desired service.
At the moment the metamodels used in the DBE to model the BML and the SDL are completely independent. It's strongly suggested that a common way to define the concept of “model” and to identify the models and a common versioning method for all types of models should be defined. This subject is treated in [SDLDef].

16 Semantic Service Language. See [DBECoreArch].
17 In this case, the Java code.

9Service Manifest and SDNA

“The SDNA represents the conceptual definition of a service when it is not related to any real service. It is the pair: SDL and BML model.” [DBECoreArch] The Figure 7 - SDNA. Service Manifest, Agreement and Service Proxy Logical Model [DBECoreArch] may be useful to focus on the difference between the SM and the SDNA.

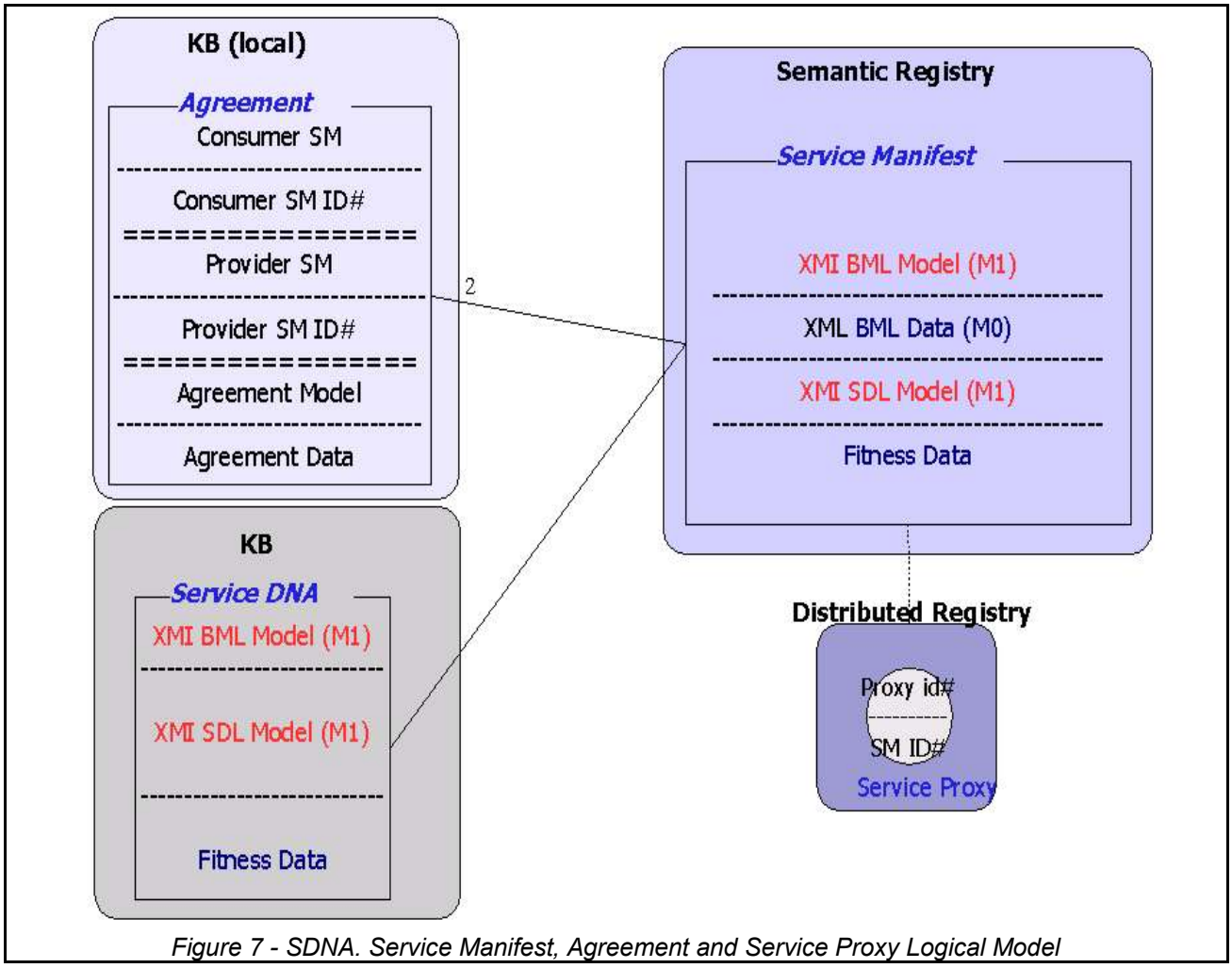


Figure 7 - SDNA. Service Manifest, Agreement and Service Proxy Logical Model

The agreement does not concern the scope of this document, refer to [DBECoreArch] for a careful exposition. In this document only SDNA and Service Manifest are discussed.
Although the vision in Figure 7 - SDNA. Service Manifest, Agreement and Service Proxy Logical Model is an high level vision and it is not updated, it's useful to underline the meaning of SDNA and to discover the differences between SDNA and SM.
The main difference between SDNA and SM is that in SDNA there are no “data”, only models (M1 in MDA term) are associated with the SDNA; the data (M0 in MDA term), for instance values for location, addresses, reference person and so forth, are to be stored in M0 part of the SM.

The SDNA role is well defined in [DBECoreArch] by this sentence: "*The SDNA is an element of reuse across services. It is the conceptual definition of a service when not related to any real service*".
The SDNA is a structure, it contains the BML and SDL Models. While the BML model represents the business and the semantic aspects of the service, the SDL model represents its technical interface.
The SDNA is really similar to SM in most of its aspects and will follow the same rules stated for the SM for life cycle, versioning and editing policy.

9.1Service DNA (SDNA)

SDNA is very similar to the SM, as SDNA is a copy inside the SM. Due to this similarity, the fields of SDNA are fully explained in the chapter 10.1 - Service Manifest Structure while further information on editing/versioning of SDNA can be found in chapter 13 - SM Editing and Versioning.
In this chapter are discussed only the peculiar aspects of SDNA. On the other hand, all concepts that are shared with the SM are explained and discussed in the SM chapter.
The structure of the SDNA is depicted in the Figure 8 - SDNA, below.

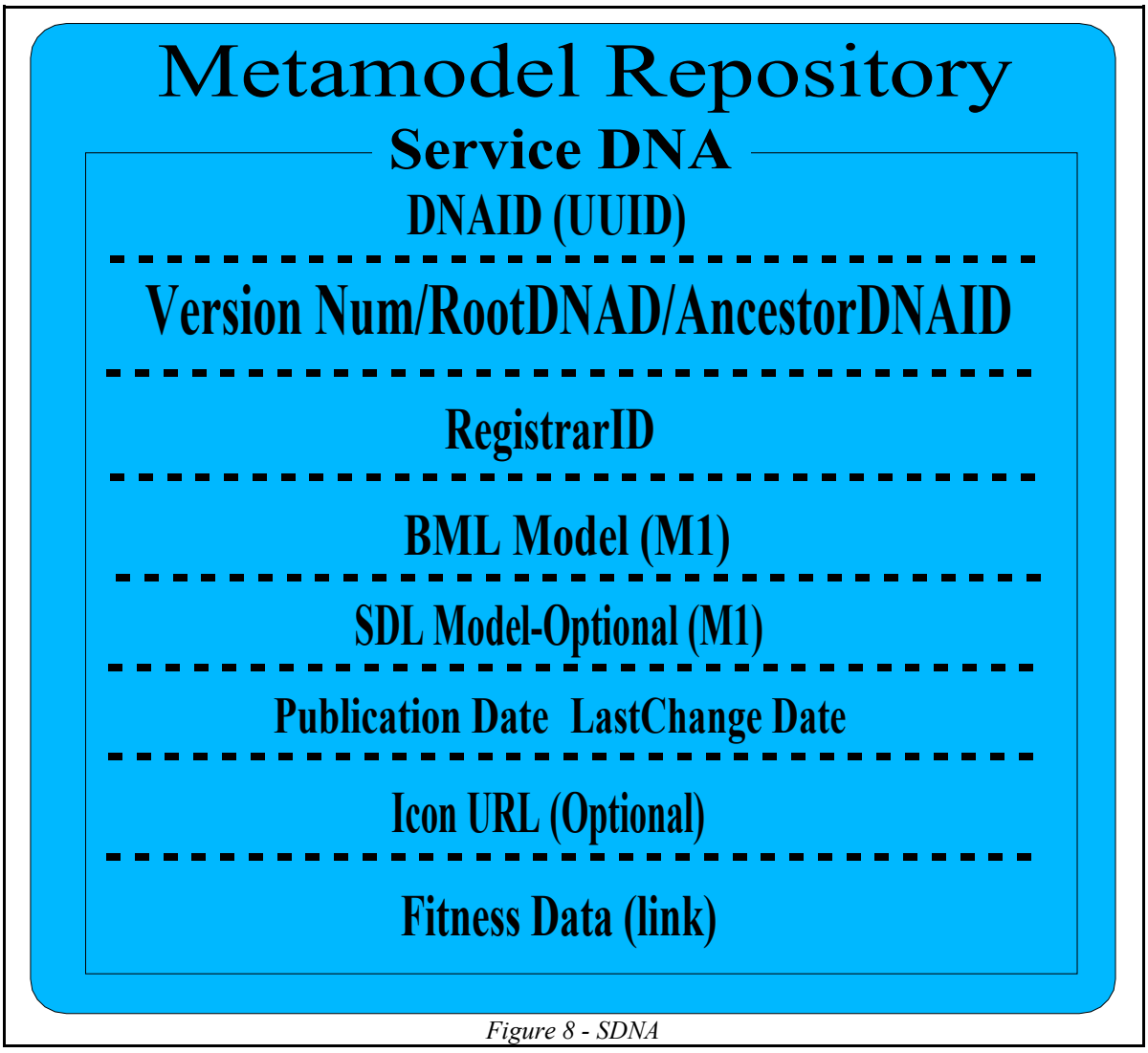


Figure 8 - SDNA

9.2Evolutionary aspect of SDNA and “standard de facto”

From the Fetish Project[18] experience it was understood that it is impossible for a commission to define a reference model of all the possible services for all business domains: this was the dream of the Diderot's “Encyclopédie Française”. It might be possible for a single business domain, as OTA[19] did, but not for an entire digital ecosystem like the DBE. For such large systems as DBE any attempt to impose a standard has very few chances to succeed, mostly because of the lack of the resources needed to impose it. Unfortunately this consideration does not reduce the importance of standards description of services in DBE, so the concept of “standard de facto” has been introduced.

The idea of “standard de facto” comes out from the evolutionary aspect of SDNA. If the SDNA a “winning SDNA” (i.e. is reused by many SM) will become a “standard de facto”, conversely, if no SM will reuse a SDNA, it will be ignored by recommendation process and will be forgotten in the KB.

The emergence of standard SDNAs is very important for the DBE environment because it eases the service composition, the recommendation process and the interoperability. The need of standard services definition is well known, the sample of OTA for travel services is emblematic.

The peculiarity of the DBE approach is the use of the “standard de-facto” idea: DBE does not aim at defining all the standard services in all business domains, but only to discover and recommend the service descriptions defined by the DBE community under the pressure of the market.

If a SDNA becomes a “standard de-facto” SDNA, its evolution will be faster than the evolution of a less used SDNA because more SM will use the SDNA and more improvements will be done on it. It's probable that from a standard DNA will rise the new SDNA standards[20].

While it is very important for DBE success to allow the emergence of some “standard” SDNA, it will be somehow “dangerous” for a SM (a service of a specific company) to become a “standard”, because it means that there is a trust and this situation may slow down the evolution of the service. This leads to an interesting analogy. In nature if an organism becomes a winner, it doesn't evolve any more because it finds easier to change its ambient rather than to change itself, but this may be another definition of the “standard de facto”. Another possible dangerous situation is when a SM overgrows and it blocks all its competitors without offering to its users further improvements, but this should be overcome by the use of open standards. As the scale-free networks show us, there will be some SDNAs that will emerge and dominate their environment while other SDNAs will still exist but will have a smaller distribution compared with the “winners”. This “points of public trust” will self emerge and even disappear according with the needs of the community which shape their environment.

18 See http://www.fetish.t-6.it/

19 Open Travel Alliance. See http://www.opentravel.org/

20 Rif. Scale Free Network. See http://en.wikipedia.org/wiki/Scale-free_network

9.3SDNA Definition

In this section the standard structure of the SDNA will be defined. The SDNA is defined using an XML Schema presented in chapter 9.3.1 - SDNA XML Schema – XSD. The other views are used only to supply a logical and conceptual vision of the SDNA, which is supposed to make it more comprehensible.

9.3.1SDNA XML Schema – XSD

The definition of the structure of the SDNA was made taking in consideration that there isn't a final definition for the metamodels of the languages involved. It was chosen to refer only to the models that will be included without constraints on their types. For example, the SDLModel reference has the anyType type and not SDLModel type. This also highlights the fact that the SM is a simple container and that it has no responsibilities on its contents. The SDNA structure is depicted in Figure 9 - SDNA XSD, below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:element name="ServiceDNA" type="ServiceDNA"/>
	<xs:complexType name="ServiceDNA">
		<xs:sequence>
			<xs:element name="DNAID" type="xs:string"/>
			<xs:element name="VersionNumber" type="xs:string"/>
			<xs:element name="AncestorSDNAID" type="xs:string"/>
			<xs:element name="RootSDNAID" type="xs:string"/>
			<xs:element name="BML_Model" type="xs:anyType"/>
			<xs:element name="SDL_Model" type="xs:anyType" minOccurs="0"/>
			<xs:element name="Availability" type="xs:string"/>
			<xs:element name="Registrar_ID" type="xs:string"/>
			<xs:element name="PublicationDate" type="xs:dateTime"/>
			<xs:element name="LastChangeDate" type="xs:dateTime"/>
			<xs:element name="Icon_URL" type="xs:anyURI" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

Figure 9 - SDNA XSD

It should be noticed that the SDL_Model attribute is optional because a service can lack a technical interface if the service was defined as a “Yellow Page”[21] service. In the Figure 10 - SDNA Schema, below the SM structure is shown in a human readable format.

21 See [DBEArchReq].

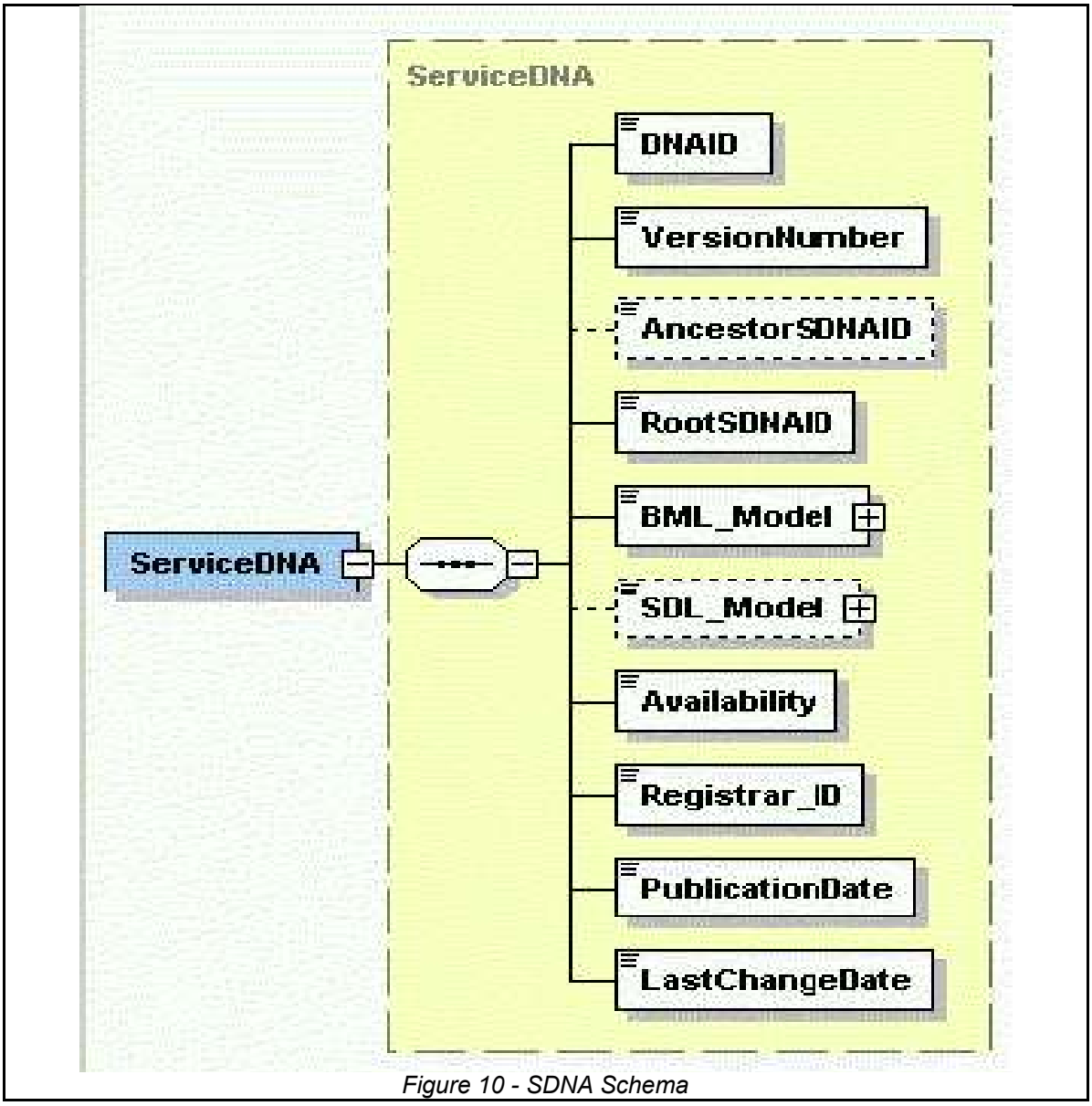


Figure 10 - SDNA Schema

10Service Manifest

Since the SDNA represents a Conceptual Service, the service manifest represents a Real Service[22]. Put it simple, a SM is “carrying” both its definition and data needed to fulfil its purpose. It contains all the specifications that describe the real service from the computing and business points of view.
The Figure 11 - Service Manifest, below shows the SM data structure and a description can be found in chapter 10.1 - Service Manifest Structure. Only the principal aspect of the SDNA is represented in the picture, not all the internal details.

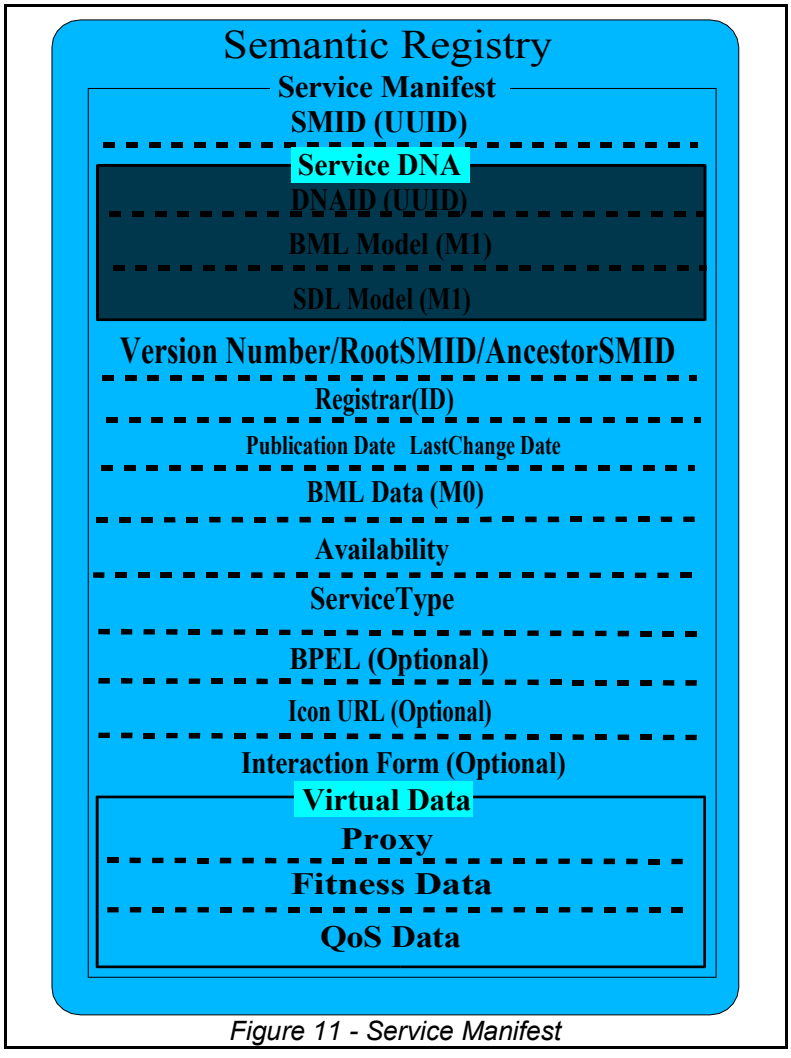


Figure 11 - Service Manifest

In the SM are stored both the models and the data that define it. However, the structure of the data section will be defined by different DBE tasks[23].
At the moment the models definition is still on-going, only the formats are assumed: while the models will be encoded in XMI 1.2/MOF 1.4, the data on the other hand will be in XML. The models used by a SM are copied from the Knowledge Base (it is a composition, not a link)

22 A service that realy exists in the real world. See [DBECoreArch].
23 For example BML Data is part of the Business Modelling Language.

and included in the SM. In this way the SM is self contained and the dependency with the KB is avoided at run-time.
Despite the current storing area of the BPEL and the Fitness Data[24] (or FD), the objective is to store the BPEL in the SM because we think that the SM have to be self contained and all the information needed to execute a service have to be contained in the SM. Where the FD will be stored is not a problem of this document, but, from the discussion with the EvE group seems that it might be stored in the EvE, the only decision that pertain to this document is that the FD will not be stored in the SM.
The Virtual Data are data related to services that are logically contained in the SM but not physically; refers to chapter 10.2 - Virtual Data for further information.
Since the SM is a XML document the natural way to define it is to supply a XML Schema. The Figure 14 - Service Manifest XSD at page 28 represents the SM in the XSD Model.

10.1Service Manifest Structure

10.1.1SMID

The SMID identifies the Service Manifest and allows the identification of the related data (proxies, fitness data, ...). To identify all the proxies which implement it, the SMID stored in the proxies itself as a foreign key is used. It's important to remember that a SM can have more Service Instances and not only one; so the SMID is not the identifier of a single proxy but the identifier of all the proxies implementing the SM. A proxy id may be used for registering/unregistering proxies in FADA. As stated above it is not possible to keep the proxy id in the SM for two reasons: many proxies may implement a single SM and the ProxyId changes every time the proxy is registered in FADA.

10.1.2SDNA

The SDNA is copied into to SM. For performance reasons this is not a link to a SDNA, but a physical copy.

DNAID

The DNAID identifies the SDNA whose SM is an instance. Every SM must contain a copy of its SDNA, not only a reference to the SDNA. The DNAID may be useful to identify all the different SM implementing the same SDNA.

BML & SDL Models

The BML Model and the SDL Model are stored in the SM, this models are copied within the SDNA. BML Model and SDL Model are stored in XMI 1.2/MOF 1.4 format.

10.1.3Version Number/AncestorSMID/RootSMID

One important SM extra functional feature is the traceability; once the SM is published all the changes are traced. To trace all the changes may be a hard task and it might be useless and not in line with the requested features of the project, for that reason only some of the changes are traced. In short, only some information (the M0 level for instance) of the SM can change at run-time and no

24 How fitted is a service for a business. It is dealt by EvE.

trace of them may be useful, but if the model related information is changed a new version of the SM must be published and traced.
The problem is detailed in chapter 13 - SM Editing and Versioning.

- **Version Number:** it identifies the version of the SM. It's a free data, and it must be SME provider who decides its format and meaning. It's suggested a format like xx.yy where xx represents the major release and yy the minor release. The DBE does not impose any specific format on it, but it's suggested to change the major release when SDNA changes. Since the Version Number is only a descriptive string, it can't be used to identify the precedent version of the SM, for this purpose the AncestorSMID is used. The versioning is a common problem for the DBE languages and representation, and it is an Open Issue for this document until a common vision will be defined.
- **AncestorSMID:** represents the previously published SM version. If the SM is the first SM on the top of the chain the AncestorSMID is omitted.
- **RootSMID:** represents the first SM in the ancestors chain. It is useful to find all the SMs with a common origin. If the SM is the first SM on the top of the chain the RootSMID is set to SMID value.

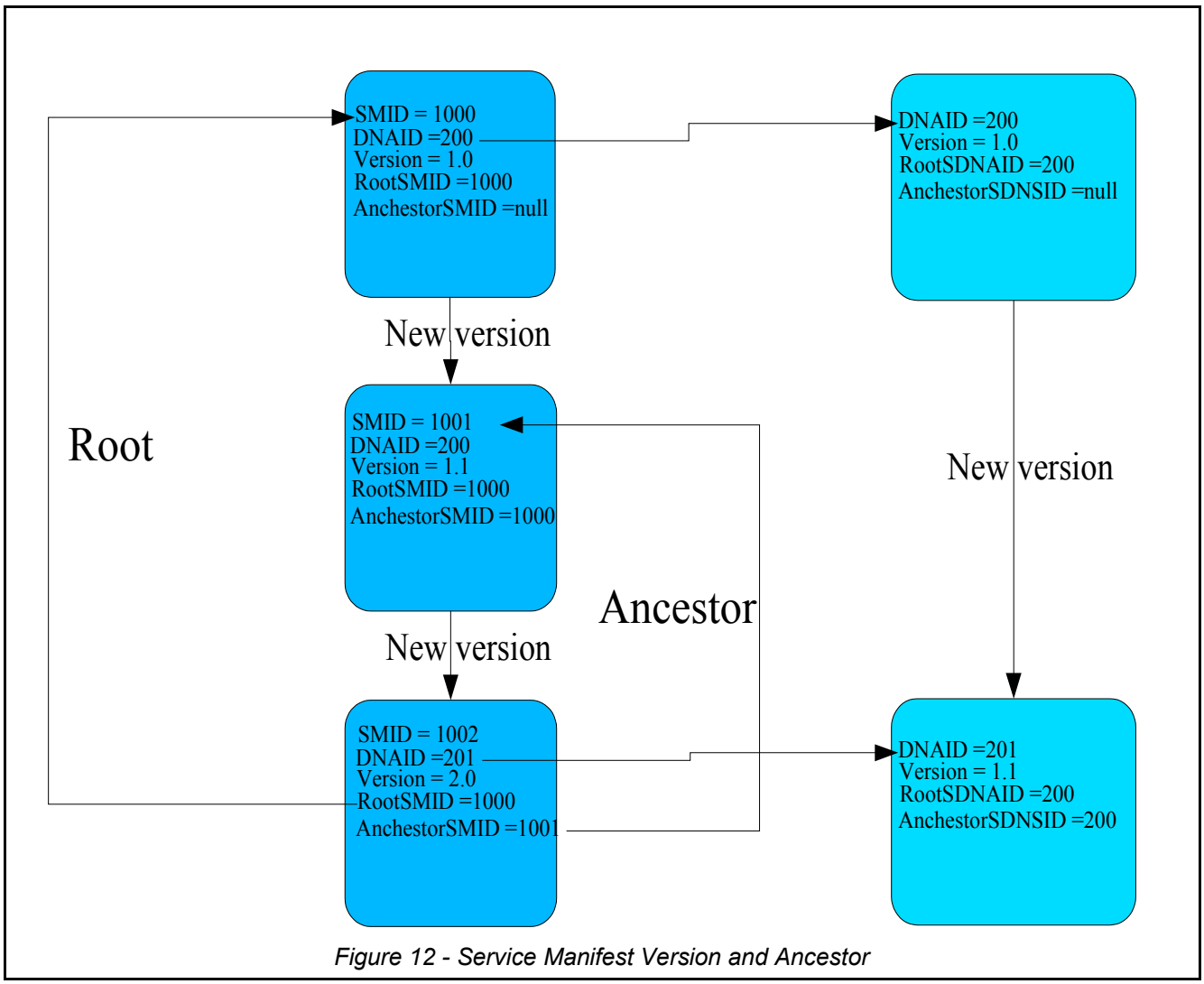


Figure 12 - Service Manifest Version and Ancestor

The Figure 12 - Service Manifest Version and Ancestor, above shows the relationship between Version Number, AncestorSM and RootSM.

10.1.4Publishing date

The date of the publishing of the current version of the SM.

10.1.5Last Change Date

The date of the last change of the SM. This date may differ from the Publishing Date when the changes to the SM don't involve the publication of a new version of the SM.

10.1.6RegistrarID

The RegistrarID aims at identifing the publisher of the service. Only the registrar can modify the SM, a security procedure to identify the registrar and to stop unauthorized changes, has to be supplied by the SR.

10.1.7ServiceType

The ServiceType defines the type of the service. The identification of the service type is needed at run time, for instance to distinguish a Business Service from an “Yellow Page”.
Up to now the types of services defined are (from [DBEArchReq]):

- Business Services:
  - “Computational Interface”
    - Simple Service
    - Composed Service
  - “Interaction Form”
    - Simple Service
    - Composed Service (Only with one Basic Service)
  - “Yellow Page”
- Basic Services
- Structural Services

10.1.8SM Availability/State

As stated in chapter 11 - Service Manifest Life Cycle the SM cannot be removed from the SR even if its Service Instance is no longer in use. There is, then, the need to discriminate the Available services from the services that are “dead”, or from the services that are still supported but only for the ongoing agreements and they will be no longer supported in the future. For these reasons the Availability/State of the service has been introduced.

10.1.9BML Data

The BML data stores the information defined by the BML Model. These information are in XML format[25].

25 Refer to [BMLMM] for further information.

10.1.10Interaction Form

The interaction form is the representation of the User Interface shown when an "Interaction Form" service is executed. The Interaction Form may be composed with a Basic Service [DBEArchReq] (Send Fax, Send Mail, ...) to perform simple operations; in this way the SME can supply a simple service without writing code or possessing an IT system.
For more information about Interaction Form refers to [DBEArchReq].

10.1.11BPEL Model

The storage policy for the BPEL has been defined after a long discussion. In the beginning it was decided to store the BPEL in the KB, but the suggestion to store the BPEL in the SM, as stated in this document, has been accepted by the DBE partner directly involved in BPEL use or definition.
From its early versions, it was suggested by this document to consider the following point:

- The BPEL Model is optional (is used only for the Composed Services)
- The BPEL Model may contain Intellectual Property information and hence to be considered as non-disclosure
- The BPEL Model may be public (to advertise the service)
- The BPEL Model is needed at runtime to execute the service
- The SM has to be Self Contained[26]

For the first implementation had been decided to store the BPEL in the KB but in the future implementations the BPEL will be part of the SM.

10.2Virtual Data

The virtual data are information that are not contained in the SM, but are universally recognized by the DBE community as "SM stuff". Since the SM is the "representation of a service" all the data related to a service are emotionally felt to be part of the SM. Still, the SM may not contain such data nor a reference to it, instead it should be the virtual data who contains a reference to the SM. For instance in the [SMCM] the Proxy and the Fitness Data are supposed to be part of the SM: this may appear as correct from the business point of view[27], but it is incorrect from the technical point of view[28]. By introducing the concept of Virtual Data we are dealing with this fact, i.e. these data are not contained in the SM but only related to it. To implement this relationship the SMID is used.

26 The SM have to contain all the data needed to define and run a service. For performance reasons is not suggested to physically distribute the information in different storage support.
27 CIM
28 PIM

10.2.1Fitness Data

The Fitness Data is not needed in order to execute the service, for this reason it will not be stored in the SM. The Fitness Data structure is an unknown specification at the moment. Anyway it's preferred to have a conceptual link to Fitness Data, wherever it will be stored, because this data may change frequently and are not accessed during the consumptions phase of the service, furthermore this information might grow considerably over time. For further discussions about the Fitness Data refer to chapter - .

10.2.2Quality of Service (QoS)

As stated for the Fitness Data the QoS[29] data are not stored in the SM. The SM has just a conceptual link to the QoS. Refer to chapter - for more information about QoS.

10.2.3Proxy

The SM does not contain the proxy and in addition it does not contain a link to the proxy. The link between the SM and the proxies (remember that a SM may have more than one proxy and a proxy might implement more than one SM) is made when every single proxy is registered to FADA. It is the proxy that contains a link to the SM.

10.3Information not Included in the SM

This chapter is pointing on the data which somebody can think that should exist in the SM structure, but for vary reasons it wasn't included. For instance, the registrar data and the textual description of the service aren't included in the SM.
The registrar data are administrated by a dedicated module which deals with the security/identification and who has nothing to do with the description of the service.
The textual description of the service, instead, is in contradiction with the DBE logic which expects that the description of the services is computational, and not descriptive. This is because the search for services should be done automatically by the recommender. However, to the end-user the description of the service, expressed using BML, should be presented using an understandable and comprehensive format. Therefore, it wasn't considered useful to duplicate such information by including textual descriptions in the SM.

10.4Service Manifest Definition

Figure 13 - Service Manifest Schema represents the current version of the SM schema. The following XML schema contains only the structure for a service, as it has been defined so far, and it doesn't formalize the aspects analysed in chapter - .

29 Data which define the quality of a service.

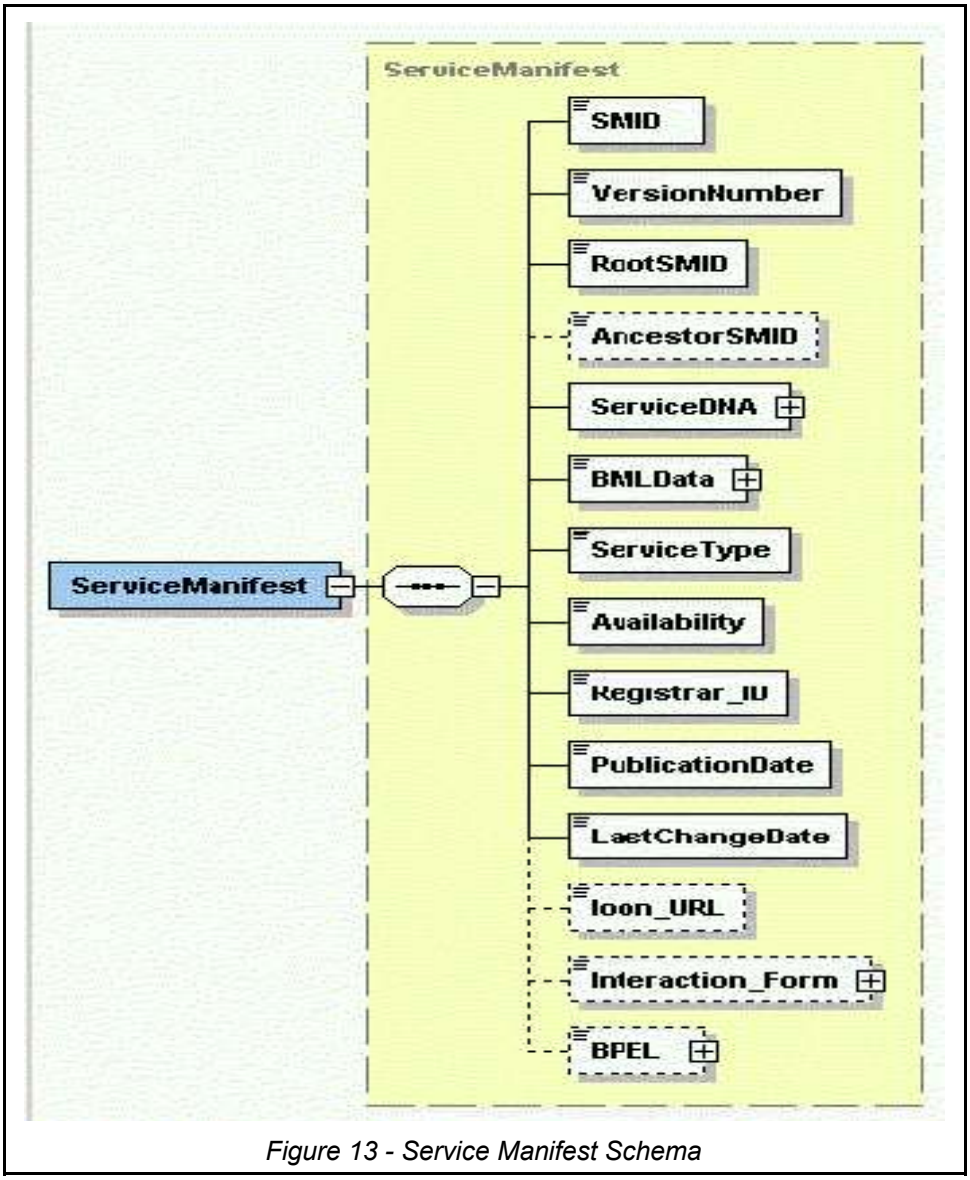


Figure 13 - Service Manifest Schema

More details will be added once there will be a better definition of the SM features and the features will be accepted by all DBE partners.
In the first version of the SM (00.02) the XSD should define also the SM contents (like SDL, BML etc) in order to provide a "validation" method for the SM and its contents.
Now it's suggested to consider the SM only as a container that has no responsibility on its contents and the XSD as a way to represent only the SM structure, but not the structure of the SM contents. If the SM stored also the BML and SDL definitions it would change when the SDL and the BML are changed.
The XSD is used to emphasize that SM is not a model or a metamodel but only a container of data, some of which might also be models. The Figure 14 - Service Manifest XSD contains the SM definition.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:element name="ServiceManifest" type="ServiceManifest"/>
	<xs:complexType name="ServiceManifest">
		<xs:sequence>
			<xs:element name="SMID" type="xs:string"/>
			<xs:element name="VersionNumber" type="xs:string"/>
			<xs:element name="RootSMID" type="xs:string" minOccurs="0"/>
			<xs:element name="AncestorSMID" type="xs:string" minOccurs="0"/>
			<xs:element name="ServiceDNA" type="ServiceDNA" minOccurs="0"/>
			<xs:element name="BMLData" type="xs:anyType"/>
			<xs:element name="ServiceType" type="xs:string"/>
			<xs:element name="Availability" type="xs:string"/>
			<xs:element name="Registrar_ID" type="xs:string"/>
			<xs:element name="PublicationDate" type="xs:dateTime"/>
			<xs:element name="LastChangeDate" type="xs:dateTime"/>
			<xs:element name="loon_URL" type="xs:anyURI" minOccurs="0"/>
			<xs:element name="Interaction_Form" type="xs:anyType" minOccurs="0"/>
			<xs:element name="BPEL" type="xs:anyType" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:element name="ServiceDNA" type="ServiceDNA"/>
	<xs:complexType name="ServiceDNA">
		<xs:sequence>
			<xs:element name="DNAID" type="xs:string"/>
			<xs:element name="VersionNumber" type="xs:string"/>
			<xs:element name="AncestorSDNAID" type="xs:string"/>
			<xs:element name="RootSDNAID" type="xs:string"/>
			<xs:element name="BML_Model" type="xs:anyType"/>
			<xs:element name="SDL_Model" type="xs:anyType" minOccurs="0"/>
			<xs:element name="Availability" type="xs:string"/>
			<xs:element name="Registrar_ID" type="xs:string"/>
			<xs:element name="PublicationDate" type="xs:dateTime"/>
			<xs:element name="LastChangeDate" type="xs:dateTime"/>
			<xs:element name="loon_URL" type="xs:anyURI" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

Figure 14 - Service Manifest XSD

10.4.1SM Class Diagram

The SM class diagram is useful to understand the logical composition of the SM, and it is depicted in Figure 15 - Service Manifest Class Diagram, below. The SM XML Schema was generated from this class diagram. Note that all the associations are compositions, i.e a copy of SDNA is stored into the SM, there are no links. The same consideration is valid for the associations between the SDNA and SDL/BML Models.

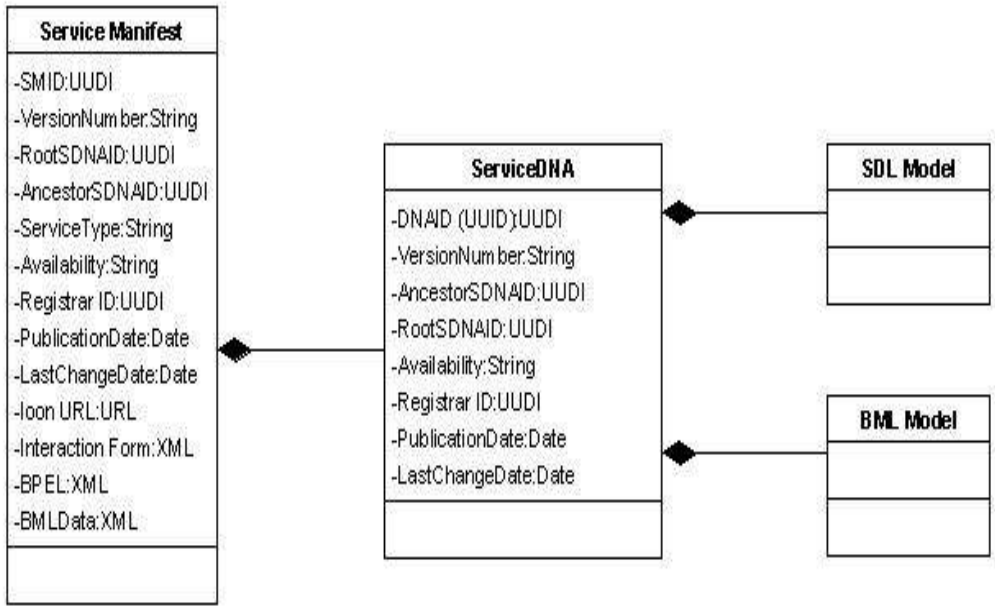


Figure 15 - Service Manifest Class Diagram

11Service Manifest Life Cycle

One of the most interesting features of FADA is that if a service is down, its proxy is removed in a couple of seconds depending on a configuration value. This allows to search only proxies of active services. As a consequence, the life cycle of a proxy is the same of the back-end service.
Now the question is: does the SM and the Proxy have the same life cycle ? Moreover: why is this choice important?
We suggest that the availability of the Proxy does not directly impact on the life cycle of the SM, i.e. the SM has to be still reachable even if its proxy is not. The different Life Cycle between SM and proxy can also allow the chaining of services with the Composer if, at design time, the proxy is down. For instance, in case a SME has its Information System active only during the office hours, the Service Proxy will be active only 8 hours in a day, but the SM will be reachable all the day long.
If the proxy is not continuously present, it will impact some qualitative parameters (presumably the uptime or the availability) to underline the irregularity of the service. As a suggestion, the Servent should be able to understand the different reasons for the proxy to be removed. It can be either for an unexpected network failure or for an explicit command. In this way the QoS (Quality of Service) information can infer the right reasons for the proxy unavailability and evaluate in a different way the two types of unavailability.
Another consideration is that it may be highly time consuming to check through all the network if at least a proxy is still active, then this operation can't be frequently performed. Note also that there might be services (informative or “Yellow Pages”) for which a service instance is not present, in this case it doesn't have sense to bind SM life cycle to the proxies one's.
For all these reasons it's strongly suggested that the SM to be removed independently from proxies.
The Figure 16 - Service Manifest Life Cycle, below shows the SM life cycle in terms of states and transactions.

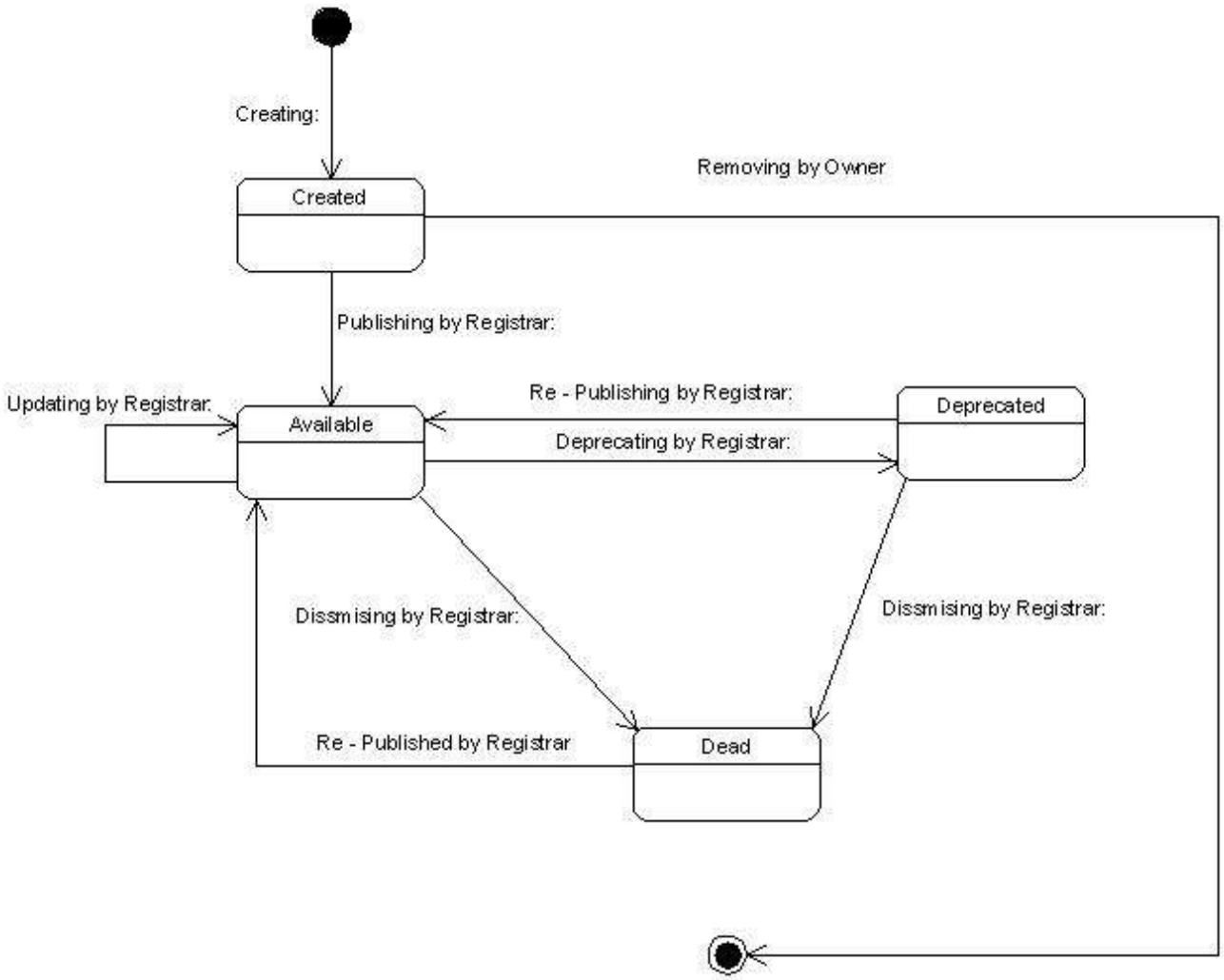


Figure 16 - Service Manifest Life Cycle

11.1SM's States:

- **Created**: the SM is private to the owner and is not shared with the DBE community. It is stored in the DBE Desktop. The owner can modify the SM without restriction.
- **Available**: the SM is shared with the DBE community or groups of users. The changes to the SM are subjected to the rules defined in chapter 13 - SM Editing and Versioning.
- **Deprecated**: the SM still remains in the SR but it is not possible to have new agreements or to compose it in a new service chain. The SM Owner must guarantee the availability of the related proxy until the end of all the ongoing agreements. In any case the SM can be used as an ancestor for the creation of new SM. The registrar may re-publish the service.
- **Dead**: the SM has no proxies available. The SME provider has decided not to support the service any more. There aren't ongoing agreements. The SM remains in the SM for historical purpose.

11.2SM's Transactions

- **Creating**: the SM is created by the owner using the SF, it is private to the owner[30] and not shared with the DBE community.
- **Publishing**: the SM is published (registered in the Semantic Register) by the registrar[31]. After the publishing the SM becomes public and only the registrar can change it.
- **Updating by the registrar**: if the SM is updated it still remains Available in the SR. The SM may be updated only by the registrar.
- **Dismissing by the registrar**: the registrar can dismiss the SM. It still remains in the SR but it isn't used. The registrar dismisses the SM when he won't maintain the service any more.
- **Re-publishing by registrar**:a dismissed or dead SM might be re-published by the registrar if he wants to rise again the service.
- **Removing by the owner**: the SM can be deleted by the owner before the publication. The DBE community will never know the existence of this service.

It should be repeated that a service published in DBE won't be removed from SR. There isn't a transition from the DEAD state to the final state. The only way to delete a SM is when this is in the state of Created, before it is published. It was considered to be proper to not ever delete a SM because it may be re-published, or it may be reused to create a new service, or even only for statistically purposes.

30The entity who owns the intellectual propriety of the SM.
31The person who publishes the SM in the Semantic Register. He is the only authorized to edit the SM. May be the SM Owner.

12SM scope/visibility

Even if the original intent was to make a SM available to the entire federated community It is reasonable to consider the existence of “private” SMs. An user can create a SM for personal use and storing it in the DBE Desktop, and does not want to share it with the other users because, for instance, it will give him an advantage on its competitors. This concept can be extended and a SM can be available only to a group of chosen users (its habitat) geographic space or other concepts of community. It is all to be established the impact of the rise of such business communities, and its implications: the concept of globalisation is lost for the concept of community even if not based on geographically neighbouring but on vary kinships that remain to be discovered and defined.
The idea of community can be expanded to the concept of a “SM market”, on which the SMs can be sold or bought. A possible solution to this issue can be the encryption of such “private” (or commercial) SMs which can be used only with a cryptographic key made available on arbitrary bases. It remains to be established the way in which a SM will be part of a “workgroup” and how a such “workgroup” should be defined.

13SM Editing and Versioning

The SM may change in order to reflect some changes to the real service[32] or to the instance service[33] represented. If the changes of the service pertain to the service implementation, they do not affect the SM, but if they pertain to the business policy (represented by BML) or the service interface (represented by SDL) these changes will affect the SM. When a SM changes the event has to be notified to all the DBE parts interested, that may use that SM.
The changes to the SM can happen before the publication or after the publication. While in the first case it does not represent a problem because the SM has not been published yet, in the latter case the re-publication may cause some trouble and must be done with care. To understand why to change a SM might be a problem you may remind that the SM can be used in a service chain and that the composition of services is done using the SM.
The Figure 17 - Service Manifest Changes explain the effect of a change in a SM SDL.

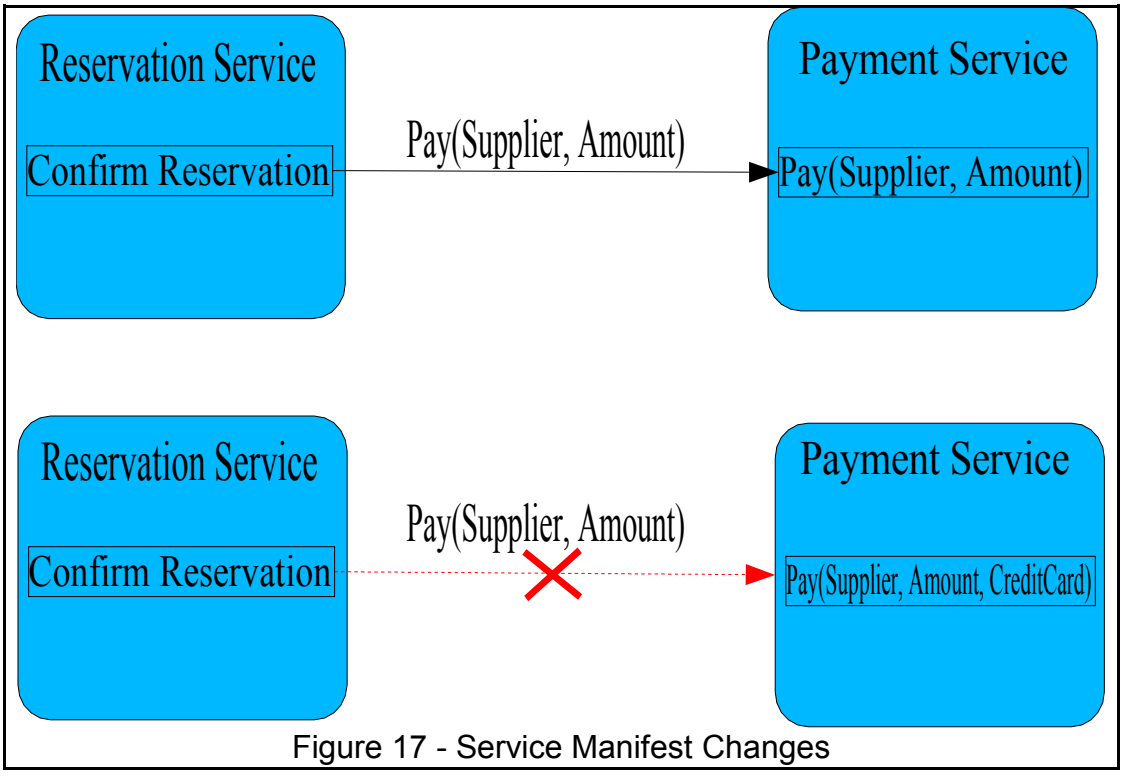


Figure 17 - Service Manifest Changes

As showed in Figure 17 - Service Manifest Changes, above the changes in SM Payment Service may produce errors in service chaining: the Confirm Reservation method tries to use the Pay(Supplier, Amount) functionality, but the functionality doesn't exist any more.
To avoid these problems the tracing and the versioning were introduced. Once the SM is published all the changes are going to be traced, and if a change is done to a SM a new version of this SM has to be published.
The Figure 18 - Service Manifest Versioning, below shows how versioning may avoid the problems pointed out by Figure 17 - Service Manifest Changes, above.

32A service that exists in the real world: for example the supplier of a real service has a VAT number, a snail mail address or an IP address.
33The IT implementation of a Real Service.

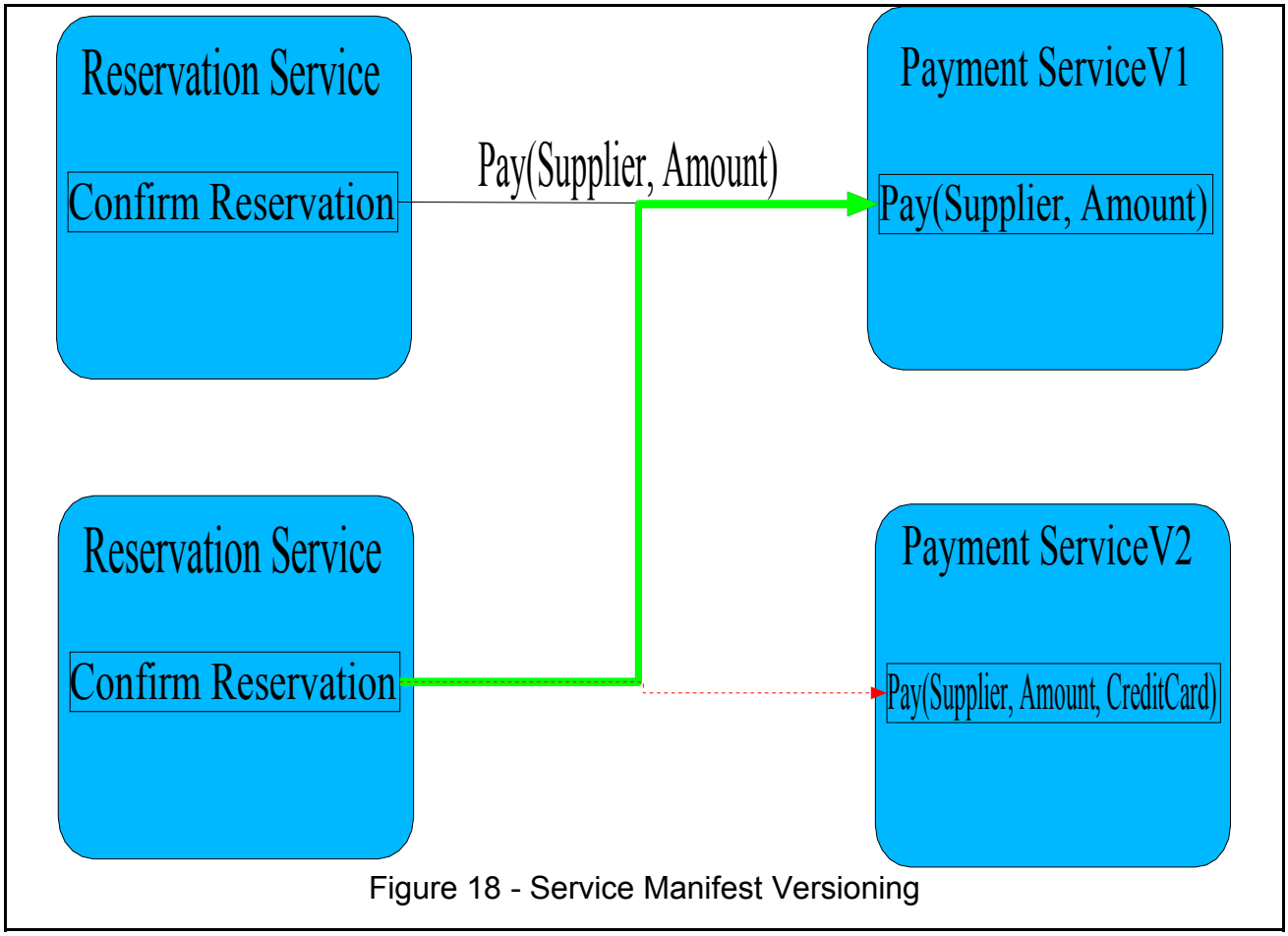


Figure 18 - Service Manifest Versioning

If a change is made on the Payment Service a new SM has to be published, then Reservation Service can use the “old” SM until its “owner” decides to modify it in order to use the new Payment Service.
It has to be noticed that the changes in SM may affect both the data and the model of the service.
The Figure 19 - Service Manifest Versioning Policy summarizes the changes to SM that imply the publication of new SM.

Attribute	*Editable*	*New Version required*
SMID	**N**	**-**
VersionNumber	**Y**	**Y**
RootSMID	**N**	**-**
AncestorSMID	**N**	**-**
ServiceDNA	**Y**	**Y**
BMLData	**Y**	**N**
ServiceType	**Y**	**Y**
Availability	**Y**	**N**
Registrar_ID	**N**	**-**
PublicationDate	**N**	**-**
LastChangeDate	**N**	**-**
Ioon_URL	**Y**	**N**
Interaction_Form	**Y**	**N**
BPEL	**Y**	**N**

Figure 19 - Service Manifest Versioning Policy

The following sections will investigate the effects of the changes to the SM.

13.1SM changes and agreements.

When an agreement is closed between a consumer and a supplier, the reference version of a the SM used, is copied into the agreement (the agreement is stored in the private storage space of the two parties), than all the future changes to the SM will not affect the on signed agreements.

13.2BML & SDL Changes

If there is the need to change BML Model or SDL Model, the changes have to be done on the Model Repository, a new version of SDNA have to be produced with the new modelsand the new version of SDNA must to be copied into SM. If the SDNA changes also the Version of the SM MUST change; this means that the SM is a new one. However, no changes are possible on the BML Model and the SDL Model inside the SM.

13.3ServiceType Changes

It's nearly impossible that only ServiceType is changed, usually with the ServiceType also other important information is modified. Anyway if ServiceType is changed a new version of SM has to be published.

13.4BML Data (M0) changes

The BML data defines the InstanceService, i.e. it contains the specific data of the service and of the service provider. If the BML Model defines that the service provider had an address and a telephone number the BML data should contain the address (1505 Piccadilly – London) and the telephone number (+44 (020) 7493 1234).
The changes to the M0 section of the SM has to be notified by the Interceptor Framework, that will be discussed in chapter 17 - . The SM Version does not have to be changed. Given the fact that when an agreement is signed the related SMs are copied in another storage area[34], any changes in the published SM will not affect the ongoing agreements unless such changes aren't agreed between partners.

13.5Interaction Form Changes

Changes to the Interaction Form doesn't force to republish a new version of the SM. If the changes only concern to the Interaction Form, it's assumed that the service isn't changed but only its User Interface is changed or has been changed. The end-user is not really affected by these changes. The changes have to be notified to the interested component, which subscribed to that service, by the Interceptor Framework, that will be discussed in chapter 17 - .

13.6Fitness Data Changes

Only the EvE changes the Fitness Data and these changes don't affect the SM nor the represented service. The changes to the Fitness Data are simply ignored. It should be remembered that the Fitness Data, as QoS Data, aren't stored in SM but there is only a conceptual link to them.

13.7BPEL Model changes

How to manage BPEL Model changes is an interesting point to discuss especially if we consider that the BPEL Model is stored, at the moment, in the KB, and its changes could be unknown to the SM. Also leaving aside technical aspects the BPEL Model changes affect the implementation of the service not its functionalities. For this reason it's suggested that changes to the BPEL Model to not impose the publication of a new version of the SM. These changes may be notified to all partners with ongoing agreements.

13.8Versioning

When we think at the SM we think at a service that evolves to satisfy the customers need. The evolutions of a SM can be very limited or can bring to a complete different service. When analysing the evolution of a service, the following considerations have to be taken into account:

- The SM has to evolve to follow the evolution of the real service .
- The business model of the service provider may change.

34 A service is stored in a local storage area for later retrieval for successive execution

- The implementation of the service may change.
- All the alteration of the SM must be traced and notified to all who are interested.
- The Registrar (the only that can modify a published SM) when try to change the SM has to be identified (for instance by his signature) as suggested in chapter 14.1 - Service Manifest Security.

The new SM version has a different SMID and a different Version Number, the RootSMID doesn't change and is used to identify all the SM versions “descending” to the same root SM. AncestorSMID is used to identify the “father” of the SM. If a SM has an empty anchestorSMID and the RootSMID = SMID the SM is the first version of the SM.

14Semantic Registry

This document doesn't aim at discussing the Semantic Registry in its internal details, its main goal is to investigate how the SR stores the SM. We suggest that the SR implementations and features to be influenced by this document.
In this chapter some SR requirements will be explicitly enlightened from the SM viewpoint.

14.1Service Manifest Security

The service manifest has to provide integrity, message authentication and signer authentication. Security is a responsibility of a DBE specific task (WP22 - DBE Security and Trust) so this document aims at suggesting some possible Security approach for the SM also if this task was postponed (after the 3rd year).
Since the SM is an XML document, the security may be implemented using the XML-Signature Syntax and Processing[35]. Since the SM may reuse the BML and the SDL Model, it is suggested that the signatures are enveloped in each single components. This can guarantee the integrity and the authentication of every single component.
The entire SM may also be signed to guarantee the overall integrity.

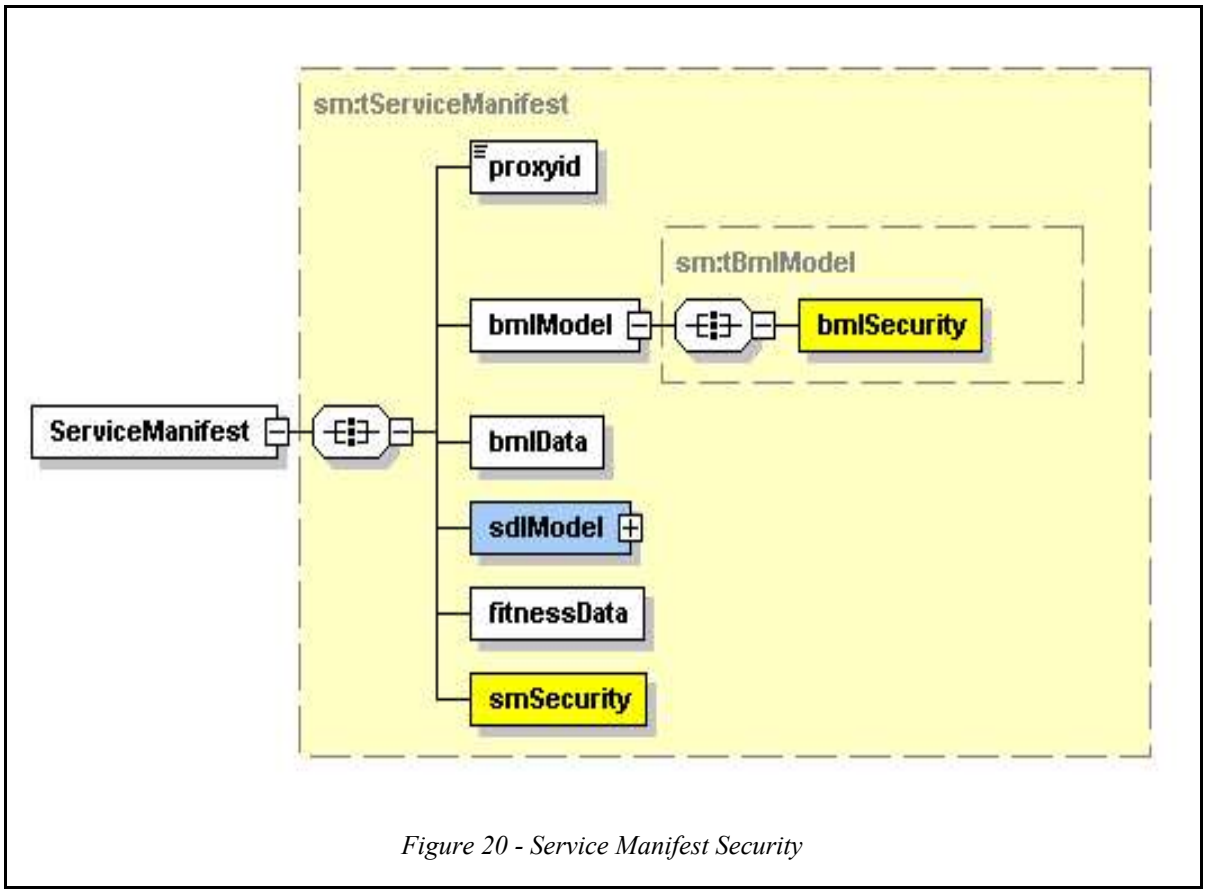


Figure 20 - Service Manifest Security

The Figure 20 - Service Manifest Security, above shows how the signature may guarantee that the bmlModel and the SM has not been tampered. If a supplier A builds

35 A WC3 recommendation. See [XMLSig].

a SM reusing a bmlModel of a supplier B, the signature of the bmlModel guarantees that the supplier A did not modified the bmlModel.
The signature of the SM guarantees the authentication of the entire SM, i.e. that no one has modified it.
The XML Signature recommendation supports the X509 standard certificate. There are a lot of FLOSS Java libraries implementing it: for those reasons it's suggested to use the XML-Signature recommendation of the W3C.
Since the SM is divided into two main parts, SDNA (BMLModel, SDLModel, SSLModel) and Service Instance (BMLData and SMID) and since lot of SM may have the same SDNA, the overall SM signature is split in SDNA Signature and the SM Signature. This allows to have two different signatures for the Model (SDNA) and the Data.
The SM would need Fitness Data to be signed by the DBE because it is a sensible information provided by the DBE EvE. It's important to remember that Fitness Data are changed at run time by the EvE, those data can not be signed by the service provider at design time because the Fitness Data are not yet available. The Fitness Data have to be signed every time they are changed by DBE EvE component. Remember that the SM has only a virtual link to Fitness Data and it is not necessary to change SM signature when Fitness Data changes because the SM doesn't really change: it's only the virtually linked data that changes.

14.2Tracing

As stated in chapter 10.1.3 - Version Number/AncestorSMID/RootSMID all the changes in the SM has to be traced. The trace is useless if there isn't any mechanism that allows to navigate through the different versions; for instance, the SM needs the SR to allow to retrieve all its new versions. Minor changes, i.e. the changes that does not imply a new version of SM, are non stored by the SR, if needed they might be stored in the DBE Memory.

14.3SM Browse and Discover

The SR has to supply some API to search the SMs as well as some visual tools to browse and discover them. The SM research have to be based on:

- SDNA
- models semantic
- models features
- SM Data
- SM ancestors/descendant

14.4CRUDEL[36] Operations

The SR has to supply some APIs to perform the basic CRUDEL operations taking into account the security and the editing rules stated in this document.

36 It's an acronym for: Create, Retrieve, Update, Delete, Exists, and List.

14.5SMID Creations

As stated in chapter 10.1.1 - SMID the SMID may be an UUID. To guarantee the uniqueness of the UUID and to have a common UUID creator the task might be performed by the SR.

15Service Manifest, EvE and Fitness Data

This chapter doesn't aim at discussing the internal structure of the EvE or Fitness Data but only to underline some aspects of EvE and FD that are related to SDNA and SM.

15.1SDNA, SM and Fitness Data

For our goals the Fitness Data might contain information about the usage history of a service by an user or a community. The goal of this paragraph is to highlight the difference between the “service” as represented by the SDNA and the “service” as represented by the SM. Since the SDNA represents the Conceptual Model of a service, the SDNA Fitness Data refers only to the conceptual model usage and might monitor, for instance, the number of SM “implementing” the SDNA and the number of daily access to ***ALL*** the SMs “implementing” the SDNA. On the other hand, the SM Fitness Data refers only to the usage of a single Service Instance represented by an SM and might monitor the customer's satisfaction referred to a determinate proxy.

For these reasons it's suggested to consider two kinds of Fitness Data: one for the SDNA and one for the SM. Please notice that this suggestion affects the logical data not the physical data: the SDNA FD might be an aggregation of SM FD. This suggestion aims at underlining the evolutionary aspect of the conceptual service (represented by the SDNA) as well as the service instance (represented by the SM).

The choice of the storage area for the Fitness Data wasn't an easy one. After a long period of reflection, it was decided that the FD wont be stored in the SM nor in the KB, but somewhere in the EvE. The reason is because the Fitness Data are not related only to a SM but they are specific for the use of service by an user of a group of users. However this decision doesn't imply that it isn't a relationship between the SM and the FDs. In [SMCM] it is described how the FDs are considered to be logically part of the SM. This logical link exists even if physically the FDs are stored in the EvE and not in the SM. Because the FDs are stored in EvE, any EvE can contain data which refer the same SM and which reflect the use of the SM in relation with that specific EvE.The Usage Data represents general data with global visibility which can be used by anybody who needs data about services. The EvE and the Recommender will have their own private data (perhaps simple aggregations of Usage Data). It is possible that data stored into the DBE Memory can be divided in Usage Data and Service History. And, in turn, the data of Service History can be divided in Migration History, Service Chain History, Service Evolution etc.

16Service Factory

This document doesn't aim at discussing the Service Factory (SF) in its internal details, it's main goal is to investigate the SF interaction with the SM. We expect that the SF implementations and features are influenced from this document.
In this chapter some SF requirements will be explicitly enlightened in the SM point of view.

16.1SM Publishing and Editing

The SF has to supply graphical tools for editing and publishing the SM and also for re-publishing the modified SM. The SF will use the SR API to publish the SM but it has to supply the GUI, or a link to the SR GUI, to facilitate the publication of a SM.

16.1.1Notify the SM publishing and Editing

When the SM is published or changed the SF has to notify the event to all components interested. Certainly the DBE Memory, EvE, and the QoS are interested. The notification mechanism should be the same used for the notifications of the Usage Data. Then, it should be done by the Interceptor Framework by a Publish&Subscribe mechanism. Because even the SF can be seen as a DBE service, the changes of a SM can be seen exactly as the Usage Data of a Structural Service.

17SM and Interceptor Framework

The Interceptor Framework (IF) and the SM have two main relationships: the IF has to intercept and “publish” all the events related to the life-cycle of a SM and to the usage of the services. All the legal aspects about the privacy of the information are not considered by this document. The concept consists in that all that happens in DBE is intercepted by the IF and selectively published to those components which subscribed to the events. These components will then decide what to do with the data, to use it or to ignore it.

17.1SM Lifecycle data

When a SM is published or modified the Interceptor Framework has to rise the appropriate event to notify that the SM has been published or changed. The event might be caught by the DBE Memory that stores the related data. The data stored in the DBE Memory may be used by the EvE, the ExE and by all the components that have to know the history of the services. May be that also the EvE or the recommender should subscribe the event published by the IF to evaluate in a specific way the SM Lifecycle events.

17.2SM Usage data

When a service is executed by a customer the IF has to rise an event to notify that the specific service has been used by a specific customer with specific data. The execution, or the failure, may be traced by the DBE Memory. The usage data have to be shared with all the DBE components: the EvE might obtain some Fitness Data, the Recommender some QoS and so on.

17.3SM and Agreement

The agreements can be fully traced, from those completed to those who didn't finished well. It may be useful a parameter to define the QoS.

18OPEN/COMMON ISSUES

18.1Versioning

The version for the SM is only a suggestion. Is strongly recommended that all the models and metamodels in DBE will use the same versioning policy.

18.2BPEL and Semantic composition

The BPEL is seen as a “technical” composition of services. Should the SM also contain semantically a composition of services? If the BPEL doesn't do it, who will do that? Will BPEL have the PIM and the CIM in its composition?

18.3Service DNA

18.3.1Availability/Scope

It may be considered appropriate to use for the SDNA the concepts of scope/visibility and availability/state, or to consider the SDNA always available/visible for all DBE users.
TBD

18.3.2Fitness Data

It is suggested to implement not only for SM but for SDNA too the concept of Fitness Data for each SDNA. At the moment the EvE was proposed only for the Instance Service.

18.4Service Manifest

The concepts of *habitat* and *visibility* are discussed in chapter 12 - SM scope/visibility. However, the visibility of the service is still in discussion.

19Glossary

Term	*Acronym*	*Synonymous*	*Description*
Business Modelling Language	BML		
Basic Service			Basic Services are DBE services that pertain to the following categories: payments, information carriers. The list could increase further on [DBECoreArch].
Business Process Execution Language	BPEL		BPEL is a language for the formal specification of business processes.
Business Service			The actual service supported by the DBE that is not either structural or basic
Computational Independent Model	CIM		
Conceptual Link		Abstract Link, Virtual Data	It means that the object has to store some data in order to allow external processes to get the related data. This is done in a CIM perspective. This doesn't imply that the data are stored in the object nor the object has a link to the data, but only that the data are reachable/obtainable for the object.
Conceptual Service			A conceptual service is the description of a service that is not related to a service instance.
CRUDEL			Create Read Update Delete List
Data Integrity			The property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner [ISG].
DBE CA			Certification Authority of DBE
DNA Identifier	DNAID		Unique identifier of SDNA
Enveloped Signature			The signature is over the XML content that contains the signature as an element. The content provides the root XML document element. Obviously, enveloped signatures must take care not to include their own value in the calculation of the SignatureValue [XMLSig].
Enveloping Signature			The signature is over content found within an Object element of the signature itself. The Object (or its content) is identified via a Reference (via a URI fragment identifier or transform) [1]
Fitness Data			All the data needed to measure the success of a service in an environment.
FLOSS			Free/Libre and Open Source Software.
Interaction Form			The resulting output of the Form Designer. The resulting GUI form is to be used for executing/consuming a service.
JAD Session			JAD means Joint Application Development is a methodology that involves the end user in the application's development. A JAD Session is a meeting of a discussion group.

Term	Acronym	Synonymous	Description
Message Authentication			A signature should identify what is signed, making it impracticable to falsify or alter either the signed matter or the signature without detection [DSG].
Platform Independent Model	PIM		
Quality of Service Information	QoS		Information about the technical Quality of the service like “average number of active proxy”, “no proxy available time”
Real Service			“By Real Service we mean a service that exists in the real world: for example the supplier of a real service has a VAT number, a snail mail address or an IP address.” [DBECoreArch]
Semantic Registry	SR		Part of DBE KB that store semantic information about services. (Da rivedere/prendere da TUC)
Service DNA	SDNA		It is the pair: SDL and BML model. It represents the conceptual definition of a service when it is not related to any real service. . The binding defines the tangible relation with a real service. The SDNA is an element of reuse across services. It is the conceptual definition of a service when not related to any real service; it represents an element of reuse (aka *Service Definition*).[DBEArchReq]
Service DNA Identifier	SDNAID		SDNA Identifier
Service Instance			Is the IT implementation of a Real Service.
Service Manifest	SM		All the specifications that describe an instance of a real service from the computing and business point off view.
Service Manifest Identifier	SMID		Service Manifest IDentifier
Service Manifest Owner			The entity who owns the intellectual propriety of the SM.
Service Manifest Registrar			The person who publishes the SM in the Semantic Register. He is the only authorized to edit the SM. May be the SM Owner.
Service Usage Data	SUD		When a service is used by a customer the event is logged in the DBE Memory for further usage.
Signer Authentication			A signature should indicate who signed a document, message or record, and should be difficult for another person to produce without authorization [DSG].
Standard “de facto”			An extensive consensus on a particular choice which has not been ratified by any official standards body, but which has a large market share.
To Be Developed	TBD		Areas in the document which will be developed in the future versions of the document.
Value Added Service	VAS	Service Chain	A complex service composed by many simple services.

20References

AOM: Brian Foote and Joseph Yoder, Metadata and Adaptive Object Models, 1998
BMLMM: ISUFI, BML Metamodel Ver. 2.0.1, December 2004, ISUFI
DBEArchReq: Pierfranco Ferronato, DBE Architecture Requirements Ver 2.2, August 2004, Soluta.net
DBECoreArch: Pierfranco Ferronato, DBE Core Architecture Ver 01.3, 2004/06/04, Soluta.net
DSG: Digital Signature Guidelines, http://www.abanet.org/scitech/ec/isc/dsgfree.html
ISG: Internet Security Glossary, http://www.faqs.org/rfcs/rfc2828.html
KBDI: TUC/MUSIC, Knowledge Base Design and Implementation Status Ver 0.1, October 15, 2004, TUC/MUSIC
MDA: David S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, January 10, 2003
MOF: Meta-Object Facility, http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF
SDLDef: Giulio Montanari, Service Description Language SDL definition v00.02, February 2005, Soluta.net
SMCM: Claudius Masuch, Thomas Kurz, Giulio Marcon, Service Manifest Conceptual Model, 15th March, 2005, Salzburg - Austria
XMI: XML Metadata Interchange (XMI), http://www.omg.org/technology/documents/formal/xmi.htm
XML: Extensible Markup Language (XML), http://www.w3.org/XML/
XMLSig: XML-Signature Syntax and Processing- W3C Recommendation 12 February 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

- end of document -