D11.2

Service concepts, models and method:
Model Driven Service Engineering
M12 issue

Document Owner: Y. Ducq (UB1), G. Doumeingts (I-VLab), C. Lieu (I-VLab), D. Chen (UB1), T. Alix (UB1), G. Zacharewicz (UB1)
Contributors:
Dissemination: Public
Contributing to: WP 1.1
Date: 15.11.2012
Revision: Version 1.0
### VERSION HISTORY

<table>
<thead>
<tr>
<th></th>
<th>DATE</th>
<th>NOTES AND COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>02.11.2012</td>
<td>FIRST REVIEW OF D11.1 AND MODIFICATION OF INTRODUCTION AND MAIN PARTS BY UB1</td>
</tr>
<tr>
<td>2</td>
<td>10.11.2012</td>
<td>MODIFICATIONS OF MANY PARTS AS TEMPLATES, MDSE ARCHITECTURE, AND DESCRIPTION OF MODELLING LEVELS</td>
</tr>
<tr>
<td>3</td>
<td>15/11/2012</td>
<td>INTRODUCTION OF THE APPLICATION AT BIVOLINO</td>
</tr>
<tr>
<td>4</td>
<td>16/11/2012</td>
<td>PRESENTATION OF ALL THE MODELS DONE USING THE SLM TOOLBOX</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### DELIVERABLE PEER REVIEW SUMMARY

<table>
<thead>
<tr>
<th>ID</th>
<th>Comments</th>
<th>Addressed (✔)</th>
<th>Answered (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exec summary to be emphasised</td>
<td>✔️ Done</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

1. EXECUTIVE SUMMARY 8
2. INTRODUCTION 10
3. PRINCIPLES AND CONCEPTS OF SERVICE AND SERVICE SYSTEM MODELLING 11
   3.1. From service to servitization 11
   3.2. MSEE Servitization Concepts 13
   3.3. From Enterprise to Manufacturing Service Ecosystem 15
   3.4. Service System and Service System Life cycle Management 16
   3.5. The Modeling of the Service System 21
   3.6. Architecture for Service System Engineering 29
   3.7. Conclusion 32
4. SERVICE SYSTEM MODELING AT BUSINESS SERVICE MODEL (BSM) LEVEL 33
   4.1. Basic service modeling concepts at BSM level 33
   4.2. Service modeling: relation between constructs at BSM level 36
   4.3. BSM service model templates 38
   4.4. BSM service modeling languages and tools: recommendations 45
   4.5. Conclusion 49
5. SERVICE SYSTEM MODELING: TECHNOLOGY INDEPENDENT MODEL LEVEL 50
   5.1. Basic service modeling concepts at TIM level 50
   5.2. Service modeling: relationships between constructs at TIM level 51
   5.3. TIM service model templates 52
   5.4. TIM service modeling language and tools and recommendation 57
   5.5. Conclusion 59
6. SERVICE SYSTEM MODELING: TECHNOLOGY SPECIFIC MODEL LEVEL 60
   6.1. Basic service modeling concepts at TSM level 60
   6.2. Service modeling: relationships between constructs at TSM level 61
   6.3. TSM service model templates 62
   6.4. TSM service modeling languages and tools 66
   6.5. Conclusion 67
7. USDL SERVICE DATA REPOSITORY 68
   7.1. Service system models vs. USDL 68
   7.2. Projections of SLM model concepts onto USDL model concepts 69
   7.3. Conclusion 72
8. MODEL TRANSFORMATION METHOD 73
   8.1. Basic concepts and principles for MDA/MDI 74
   8.2. Vertical transformations 75
   8.3. Horizontal transformations 78
   8.4. Integrated Transformation Example 82
   8.5. Conclusions 84
9. METHOD 85
   9.1. Description of the structured approach 85
   9.2. Step 1: Definition of the strategy for the servitization transition 86
   9.3. Step 2: AS IS Modeling at the BSM level, performance evaluation and diagnosis 86
   9.4. Step 3: Modeling of the TO BE for the MSEE (global and detailed) at TOP BSM level 88
   9.5. Step 4: TOBE at Bottom BSM Level 89
   9.6. Step 5: TO BE at TIM Level 89
   9.7. Step 6: TOBE at the TSM level 90
   9.8. Conclusion 90
10. APPLICATION OF THE MODELLING LANGUAGES TO A PILOT CASE OF MSEE 91
   10.1. Context of the company and servitization expectations 91
   10.2. Organisation of the information collection and modelling 93
   10.3. Presentation of the models 94
11. CONCLUSIONS 100
12. REFERENCES 102
13. ANNEX 1. MDA / MDI ARCHITECTURE 106
14. ANNEX 2. MODEL TRANSFORMATION 108
15. ANNEX 3. EXISTING RELEVANT SERVICE MODELING LANGUAGES 114

| 15.1. | GRAI Model | 114 |
| 15.2. | GRAI modeling language | 116 |
| 15.3. | ECOGRAI | 119 |
| 15.4. | POP* | 121 |
| 15.5. | EN/ISO 19440 | 123 |
| 15.6. | Blueprint | 125 |
| 15.7. | BPMN | 127 |
| 15.8. | Smart network modeling method | 128 |
| 15.9. | SmartService | 131 |
| 15.10. | USDL | 133 |
| 15.11. | WSDL | 134 |
| 15.12. | REST | 136 |
| 15.13. | BPEL | 137 |
LIST OF FIGURES

Figure 1. The Extended Product Concept, adopted from (Thoben et al. 2001) 13
Figure 2 Servitization process 14
Figure 3: Service Life Cycle 15
Figure 4: Virtual Manufacturing Enterprise 15
Figure 5: Business Ecosystem Concept 16
Figure 6. Manufacturing enterprise vs. service in manufacturing virtual enterprise 17
Figure 7. Service delivery system 17
Figure 8. Service System lifecycle phases vs. Service System life - adapted from Bernus (1995) 18
Figure 9. Class of Tools supporting Service Lifecycle Management 19
Figure 10: Virtual Manufacturing Enterprise in different phases of SLM 20
Figure 11. The structure of a system 23
Figure 12. MSEE as a system of systems 24
Figure 13: The control of a System 25
Figure 14: The decomposition of the control system 25
Figure 15: the hierarchical or temporal decomposition of the Decision System 26
Figure 16: Modelling of the different VME along the SLM 27
Figure 17: Enterprise Modelling mapped to the OMG 4 level architecture 28
Figure 18: Towards a Model Driven Service Engineering Architecture 31
Figure 19: System Lifecycle and Modelling Levels 32
Figure 20: Modeling constructs and relationships at BSM level 36
Figure 21: Links between INDESIT service case modeling concepts at BSM level 38
Figure 22: BSM level modeling constructs vs. some existing modeling Language 46
Figure 23: BSM level modeling constructs for the chosen modeling Language 47
Figure 24: BSM level modeling constructs vs. some existing IT tools 47
Figure 25: Use of template - example of ‘Process’ modeling 48
Figure 26: Service Modeling constructs and relationships at TIM Level 51
Figure 27: Map existing languages/ tools to TIM level service modeling constructs 58
Figure 28: Map of chosen languages/ tools to TIM level service modeling constructs 58
Figure 29: Modeling constructs and relationships at TSM level 61
Figure 30: Map existing languages/tools to TSM modeling constructs 67
Figure 31: USDL vs. Engineering models 68
Figure 32: USDL support in Service lifecycle phases 68
Figure 33: Projection of relevant service concepts onto USDL modeling concepts 70
Figure 34: USDL module "Participant" 71
Figure 35: MDA’s Conceptualization Levels and Transformations 74
Figure 36: Model Transformation 74
Figure 37: Possible CIM-code Vertical Transformation Process 76
Figure 38: Vertical Transformation Example 78
Figure 39: Method for Horizontal Transformation 79
Figure 40: Horizontal Transformation Example 81
Figure 41: Tuple: A conceptual knowledge mapping 81
Figure 42: Tuple: Structural knowledge mappings 82
Figure 43: ISTA 3 methodology 83
Figure 44: Integrated view on ISTA 3 and MDI horizontal and vertical transformations 84
Figure 45: The various situations and the combination of companies 85
Figure 46: MSEE method using MSDEA tools and concepts 86
Figure 47: The iterative approach of AS IS Modeling at the BSM level from phase 1 to 2 87
Figure 48: Iterative AS IS Modeling and Diagnosis at the BSM level from phase 2 to 4 88
Figure 49: The iterative approach of TO BE Modeling at the BSM level 89
Figure 50: Description of the manufacturing chain (AS is and TO BE situation) 92
Figure 51: The structured approach and models for the pilot case 93
Figure 52: The principles of process and control of service product and service system 94
Figure 53: The global activity of BIVOLINO
Figure 54: The decomposed activities of BIVOLINO
Figure 55: The design activities of BIVOLINO
Figure 56: The separation of design activities of BIVOLINO
Figure 57: The service product design activities process
Figure 58: The configurator customiser functionalities
Figure 59: The service system design activities process
Figure 60: The service system development activities process
Figure 61: The service system activities process
Figure 62: The control system of service product life cycle using GRAI Grid
Figure 63: The control system of service system exploitation using GRAI Grid
Figure 64: Reference Model for MDI
Figure 65: Method for Horizontal Transformations with Semantic Support
Figure 66: KMType hierarchy
Figure 67: Structure of Communication Mediator KB
Figure 68: Tuples instantiation
LIST OF TABLES

Table 1: Identifying relevant Service modeling concepts at BSM level 35
Table 2: Example INDESIT Service Modeling constructs 37
Table 3: Service Template at BSM 39
Table 4: Customer template at BSN 39
Table 5: Stakeholder template at BSM 39
Table 6: Partner Template at BSM 40
Table 7: Product template at BSM 40
Table 8: Functionality template at BSM 41
Table 9: Resource template at BSM 41
Table 10: Process template at BSM 42
Table 11: Decision template at BSM 43
Table 12: Decision structure template at BSM 43
Table 13: Organisation template at BSM 44
Table 14: Performance Indicators at BSM 44
Table 15: Value template at BSM 45
Table 16: Identifying relevant Service modeling concepts at TIM level 50
Table 17: Service Template at TIM 52
Table 18: Process Template at TIM 53
Table 19: Human resource template at TIM 54
Table 20: Organisation unit template at TIM 54
Table 21: Organisation template at TIM 55
Table 22: Physical means template at TIM 55
Table 23: Information template at TIM 56
Table 24: Enterprise Application template at TIM 57
Table 25: Identifying relevant Service modeling concepts at TSM level 60
Table 26: Process template at TSM level 62
Table 27: Organisation Unit template at TSM level 63
Table 28: Organisation template at TSM level 63
Table 29: Human (resources) template at TSM level 64
Table 30: Physical means (resources) template at TSM level 64
Table 31: Layout template at TSM level 65
Table 32: Layout at TSM level 65
Table 33: Enterprise application template at TSM level 66
Table 34: Classes of Model Morphisms 75
Table 35: Semantic Mismatches (based on (INTEROP NoE 2005)) 111
1. Executive Summary

The two main objectives of WP 1.1 are:

- to develop innovative concepts to describe Service Life cycle Management (SLM) in Manufacturing Enterprises by means of models, methods and tools which will be used by the business users along all the phases of the Service Life cycle
- to define a new modelling collaborative architecture, to support business-IT interaction and distributed decision making in service-driven virtual factories and enterprises based on MDA/MDI (Model Driven Architecture/Model Driven Interoperability).

This deliverable D11.2 is the release version at M12 of D11.1 delivered at M4 and both report on the work done in the research domain of modelling of service systems.

However, this deliverable contains all the work done and not only the additional work from the previous deliverable in order to have a complete version and to appreciate the final results. Indeed, this is the final version of this deliverable.

The main added materials of this release are, the improvement of the MDSEA, the improvement of the templates and the justification of the modelling languages at each modelling level and also the results of MDSEA application to a specific case study.

The initial chapter recalls the definition of Service Model, then of Servitization process. The Servitization process aims to link the business generated by selling a physical product to the business generated by selling its support services, being them an adjunct to the product offer (for example, adding maintenance services to a machine tool sold to a customer), or a pure product-related service offer (e.g. selling hours of machining, rather than the machine itself).

We focus also on the Service System (a complex aggregation of organizational-IT-physical resources coming from a manufacturing enterprise, a virtual manufacturing enterprise or a Manufacturing Service Ecosystem) which creates produces and evolves the Service Model. The concept of Service Life cycle Management (SLM) is introduced, as well as the concept of Service System Modelling and Engineering. We have defined also the Service System Life cycle management derived from ISO 15704 (2000) standard (ISO 15704) developed for PLM (Product Life cycle Management).

The principles concerned with Service Systems modeling are presented in chapter 3 and we underline the need to develop global modelling concepts as well as to suggest a new set of languages to describe these models. We recommend the adoption of System theory and System of Systems concepts for the global modelling of the various Service Systems because it corresponds exactly to the combination of various organizations to create the Service System. Concerning the paradigm to describe the models, we choose the OMG (Object Management Group) architecture with four levels introducing concepts, constructs and modeling languages. We have tried not to reinvent the wheel in the domain of modeling languages, but to adapt, improve and harmonise existing ones.

The chapter 3 describes also the Model Driven Service Engineering Architecture (MDSEA) which aims to separate the issues when designing and implementing a service and a service system and which defines three levels of abstraction, inspired by the MDA/MDI Architecture proposed in INTEROP-NoE project (INTEROP NoE (2005) Deliverable DTG3.1).

- the Business Service Model (BSM) to model the Service System at the Business level and which represents mainly the user point of view,
- the Technical Independent Model (TIM) level which models the domain components (ICT, Organization/Human, Physical Means) of the Service System based on BSM models but without taking in account the technology that will be used,
- the Technical Specific Model (TSM) level which provides the technical model of the various domain components and supports their realization.

MDSEA provides an integrated framework with modelling languages at the various levels of abstraction to support Service models and Service System design and implementation.
The relationships between the MDSEA modeling levels (BSM, TIM, TSM) and the Service System lifecycle phases (ideation, user-requirements, design and implementation) is also described. One of the important innovations in MDSEA is to define the integration between domains components at the BSM level in order to ensure that these integration aspects will be spread out at the other levels.

The chapters 4, 5 and 6 present the Service System modelling language defined at the three levels BSM, TIM, TSM of MDSEA.

The underlying principle is based on the EN/ISO 19440 standard for enterprise modelling. It includes a set of modelling constructs represented by templates and graphical notations. A template allows defining a set of attributes while remaining neutral and extensible. Graphical notations facilitate user’s understanding and involvement in the modelling process.

The models defined at the BSM level focus on the representation of the service (and of its functionalities) and of the Service System (Enterprise and Virtual Enterprise for WP1.1) capturing information on its related product, partner, customer, stakeholder, service KPIs and value, as well as on decision-making, organisation, resource and process.

The models defined at the TIM level are deduced from the BSM level models using model transformation mechanisms. The main focus of TIM modelling is to specify the three domain components: IT, Organisation/Human, Physical Means.

The models defined at TSM level are deduced from the TIM level by model transformations. The TSM principle is to reach a level of description which allows “to generate” domain components. In IT the objective is to generate software applications and services, whereas for the two other domains the objective is to support the progress status of implementation: for instance, recruitment or training of a specific human resource or procurement of a physical mean.

At each level a recommendation of languages is given.

The chapter 7 analyses the possibility to project all the information collected by the MDSEA via BSM-TIM-TSM models in USDL (Unified Service Description Language). In fact, only the service data needed for service commercialization and delivery will be mapped to USDL.

The additional data required by USDL that is not concerned by BSM-TIM-TSM models will be captured directly via USDL interfacing at the service operation phase. This phase is more detailed in D11.5.

Chapter 8 concerns model transformation from BSM to TIM and to TSM levels. Two types of model transformation can be considered: (1) model detailing with same modelling constructs, which requires adding attributes to a same construct template; (2) model mapping with new constructs, which requires mapping a set of concepts to a new set of concepts with possibly different semantics. The Model transformation method presented in this deliverable focuses on case (2) which is more difficult to perform. These transformation principles are deeply detailed in D11.3.

In chapter 9, the outline of a methodology to support the use of the proposed Architecture and Languages is presented at the end of the deliverable. We focus on a structured approach to allow modelling activities at the BSM-TIM-TSM levels in a guided and step-by-step ways.

In chapter 10, the application of MDSEA modelling languages at BSM level is presented. The initial pilot case is BIVOLINO. So, the approach and the modelling principles are presented as well as all the models for process and decisions.

Chapter 11 draws some conclusions on this research work and gives some perspectives of research and application.

The bibliography is at the end of the document, just before the various annexes.
2. INTRODUCTION

The challenge 1 of MSEE is “to make Service Science Management and Engineering (SSME) (name given by IBM to this new discipline at the beginning of 2000) to evolve towards Manufacturing Systems and Factories of the Future (FoF), in adapting, modifying and extending SSME concepts from a methodological viewpoint to make them applicable to traditionally product-oriented enterprises from an implementation viewpoint”.

To meet this challenge, the goal of the WP1.1 is first to develop innovative concepts to describe Service Model, Service System and Service Life cycle Management (SLM) in Manufacturing using models, methods and tools and then to develop components (IT, Organisation/Human, Physical Means) to support the development of services in all phases of SLM.

This development will be inspired by the approach of Model Driven Development particularly Model Driven Architecture/Model Driven Interoperability (MDA/MDI) in order to create Model Driven Service Engineering Architecture (MDSEA).

The deliverable will properly structure existing relevant modelling concepts, languages and methods according to the needs of SLM design rather than re-developing these elements from scratch.

The content of D11.2 must be considered as a release of D11.1 and as a presentation of the final results of WP1.1 which will be used by other workpackages of MSEE.

This D11.2 aims to present in particular the application of the modelling languages at BSM to a specific case study of the project. The followed approach and the obtained models are presented as well as interests in the use of these models.

This D11.2 presents also an accurate version of templates and justification of chosen languages.

However, this deliverable does not include only modifications in comparison to D11.1 but the complete work done in the task 1.1.1, task 1.1.2 and task 1.1.3 from the beginning of the project.

The structure of deliverable D11.2 is as follows.

- Chapter 3 presents basic concepts and definitions on service model, service system, Service Life cycle Management (SLM). Basic modelling rationals are presented and the principles for elaborating Service System modelling language are defined. We introduce at the end of the chapter Model Driven Service Engineering Architecture (MDSEA) based on MDA/MDI.
- Chapter 4, 5, 6 presents the service system modelling language at various level of the MDSEA (BSM, TIM, TSM). The interests of such an architecture are explained and then each modelling level is described.
- Chapter 7 discusses the use of USDL (Unified Service Description Language).
- Chapter 8 presents basic concepts and principles of model transformation method under the frame of MDSEA.
- Chapter 9 outlines the methodology that will support the use of service system modelling language and its associated SLM toolbox.
- Chapter 10 presents the models done using modelling languages at BSM in a specific use case of the project.
- Finally chapter 11 concludes the D11.2 and indicates future work and activities that remain to be done in this research domain and that will be presented in the future deliverables of WP1.1.
- Chapter 12 gives the list of references.

The chapters 13, 14, 15 are annexes sections presenting MDA/MDI, model transformation as well as a summary of existing modelling approaches that are relevant for use in association with the proposed service system modelling language presented in this deliverable.
3. PRINCIPLES AND CONCEPTS OF SERVICE and SERVICE SYSTEM MODELLING

The finality of this chapter is to propose the principles and the concepts for the modeling of Service and Service System able to develop, operate and govern it\(^1\). These principles and concepts answer the MSEE challenge 1, recalled at the beginning of this deliverable. To reach this objective we give the main characteristics of a Service and a Service System which produces the service in the domain of Manufacturing. Then the Service Life cycle Management concept is introduced. At the end of the chapter we propose the principles, the concepts, and the languages for the modeling of Service System. The model of the Service System will allow to generate IT applications and services and also to define the others components (Organisation/human and Physical Means) which will support all phases of the Service Life Cycle. This point is important to underline in the sense that modelling is often used only for Information System definition and implementation as in UML and originally in MDA.

3.1. From service to servitization

The aim of this sub-chapter is to remind the definition and the characteristics of the servitization which is the basic subject of the MSEE project. The studies and researches in the domain of Service have been mostly devoted to support tertiary sector domains (e.g. banking & finance, tourism, trade, public administration), with an obvious focus on ICT. At the end of the nineties, the concept of Service in Manufacturing appeared and the evolution from an economy of products towards an economy of services round the products became more and more important in manufacturing: this evolution is called Servitization and its most tangible effect is the development of Product Service Systems (PSS).

In order to produce this part, a lot of references were scrutinized (see the bibliography but also the deliverables MSEE D12.1 and D52.1).

3.1.1. Characterization of a service

The definition of a Service is very difficult. We have extracted from the survey of the literature we have performed, one definition we estimate significant: “Service is the application of competence for the benefit of another. So service involves at least two entities, one applying competence and another integrating the applied competences with other resources and determining benefit (value co-creation). We call these interacting entities service systems” (Spohrer et al.).

Most of the time a service is opposed to a good. The following list characterizes a service (Lovelock, 2004):

- A service is not owned, but there is a restricted access,
- Services have intangible results,
- Customers are involved in the service production process,
- Other persons than the customers can be involved in the service process as stakeholders, sub-contractors, etc…
- Quality in service is difficult to control while increasing productivity and also difficult to apprehend
- Service cannot be stored.
- Service delivery lead time is crucial.
- Service delivery integrates physical and electronic way.

\(^1\) In the MSEE project, the Service System is assumed to be a business ecosystem of manufacturing and service providers.
Since a decade, new research thinking has been emerging, trying to systematize the multi-disciplinary knowledge involved in service systems. On their web page, IBM describes service science as “a growing multi-disciplinary research and academic effort that integrates aspects of established fields like computer science, operations research, engineering, management sciences, business strategy, social and cognitive sciences, and legal sciences”. In the computer science domain, Service Oriented Architectures (SOA), have revolutionized information systems, by providing software engineers with powerful methodologies and tools for decomposing complex systems into autonomous components. The final aim of such an evolution is to support enterprise vital processes and workflows, by simple orchestrations and compositions in the hand of business specialists.

3.1.2. Product service and PSS

A product-service is often called a service supplied in addition to a product increasing its value for the customers (Furrer, 1997). We can refer also to the SUSPRONET European project (Product services in the need area “Information and Communication”, by Charter, Adams and Clark, Suspronet report (October 30th, 2004)), which gives the following definitions:

- a **product service** as a value proposition that consists of a mix of tangible products and intangible service designed and combined so that they are jointly capable of fulfilling integrated, final customer needs
- a **Product-Service System (PSS)** as the product-service including the network and infrastructure needed to ‘produce’ a product-service

A typology of product-service systems has been proposed by different authors that considers three PSS variants (Behrend et al. 2003; Brezet et al, 2001; Zaring et al. 2001):

- The first variant is **product-oriented services**. Here, the business model is still dominantly geared towards sales of products, but some extra services are added.
- The second main variant is **use-oriented services**. Here, the traditional product still plays a central role, but the business model is not anymore geared towards selling products. The product stays in ownership with the provider, and is made available in a different form, and sometimes shared by a number of users.
- The last variant is **result-oriented services**. Here, the client and provider in principle agree on a result, and there is not a pre-determined product involved, i.e. that the product is just a mean to sell a service but the customer pays for the service but not for the product. For instance, most of the mobile phone providers give the phone to sell communication time.

“An **Industrial Product-Service System (IPS2)** is characterized by the integrated and mutually determined planning, development, provision and use of product and service shares including its immanent software components in Business-to-Business applications and represents a knowledge-intensive socio-technical system”. Certainly, at the present time, the income generated by the sale of services and product-service systems sales is higher than that generated by product sales. Nevertheless, the evolution towards PSS is neither immediate nor obvious. It implies managerial and organizational changes that are often out of reach for most SMEs or even large industrial companies.

In the next section we will analyze the PSS in the domain of Manufacturing and the process to develop it (Servitization).
3.2.  MSEE Servitization Concepts

Clearly the servitization of manufacturing companies covers different levels of service provision and consequently different stages can be followed to evolve (MSEE, deliverable 52.1).

Traditionally in the manufacturing domain, we are used to consider the product as the core element of the service to customers. But due to the market pressure, it is necessary to offer to the customers more services linked to the product, either new services linked to existing products, either new innovate services linked to specific products developed around these services. An appropriate concept to link products, product related services and the needs of the users is the “Extended Product” (EP) (Thoben et al. 2001).

3.2.1. Extended Product (EP)

The Extended Product (MSEE D52.1) concept belongs to the category of Product-Service System. The Extended Product is characterized by a layer model based on manufacturing product and defining the process extensions (Figure 1). The Extended Product is a complex result of tangible and intangible components.

![Figure 1. The Extended Product Concept, adopted from (Thoben et al. 2001)](image)

The Core Product is the physical product that is offered to the market; while the Product Shell describes the tangible “packaging” of the product (e.g. Ibarria sells machine-tools and will add new sensors and devices for maintenance). Supporting Services are intangible additions, which facilitate the use of the product (e.g. adaptive preventive maintenance plans or mobility guarantees)

The resulting Extended Product would be the specific solution satisfying the customers demand. As the solution can become very complex, several business partners may be collaborating for the provision of the EP in the frame of an Ecosystem. Thus, the following aspects define the EP concept (Thoben et al. 2001):

- Combination of a physical product and associated services,
- Intangible extensions that are information and knowledge, extensive collaboration of enterprises in groups / networks to provide value adding services

3.2.2. Product+Service and Product2Service

The different stages of product and service provision are shown in Figure 2.
The first stage is the selling of a product (Tangible Product).
The second stage which initializes the servitization process and the evolution toward Product+Service, start by adding a simple service (Product and supporting service). In this scenario, the simultaneous offering of the tangible product (Core Product and Shell) extended with proper tailored services is developed. In this case, both physical products and services contribute to the revenues, their balance needs to be adaptively determined and continuous innovation of services assumes a key competitive advantage (e.g. IBARMIA will add a device on the machine-tool allowing to check continuously the running of the machine etc).
The third stage (Product and differentiating service) is an evolution of the previous one. The service is more elaborated and increases the differentiation. If we use the previous example with IBARMIA, we can propose to sell the machine plus a service which guaranties a high percentage of availability of this machine.
The fourth stage, Product2Service scenarios are in contrast sharply decoupling manufacturing of goods and selling of services, where in most cases physical goods remain the property of the manufacturer and are considered as investment, while revenues come uniquely from the services (e.g. IBARMIA doesn’t sell the machine-tools but sells hours of running of the machine-tool).

3.2.3. Service Life cycle Management

Service Life cycle Management (SLM) is a concept derived from PLM (Product Lifecycle Management). PLM is concerned with the management of entire life cycle of a product focusing on all product data relating to its design, production, support and ultimate disposal at the end of the life cycle. Similar to PLM, SLM aims at managing all service data relating to its design, implementation, operation and final disposal. The various phases are (Figure 3):
3.3. From Enterprise to Manufacturing Service Ecosystem

In this subchapter we analyze the evolution of the Manufacturing Service System along the various transitions from stage 1 to stage 4 (see Figure 2). In the servitization process the hypothesis is to start from one manufacturing enterprise. We will do a distinction between the transition from stage 1 to stage 2 and 3 on one side and the transition to stage 4 which is certainly more complex.

3.3.1. From a single Manufacturing Enterprise to a Virtual Manufacturing Enterprise

Extension of products in terms of Product+Service will concern physical products as well as the associated accessories or services. Thus, depending on the type and core competencies required to supply the associated services, it will be necessary to involve several business partners collaborating very closely towards the common goal of making the sale of the package attractive, sharing risks and resources (Figure 4).

An industrial model for collaboration to exploit the various opportunities without the implementation of a strong integration is a Virtual Manufacturing Enterprise (VME).

A VME is an organizational form that marshals more resources than it currently has on its own, using collaborations both inside and outside of its boundaries, presenting itself to the customer as one unit. It is a set of (legally) independent enterprises that share resources and skills to achieve a mission/goal. The main difference between Supply Chain and VME is the type of relation between the enterprises (more integrated in Supply Chain) but also the fact that the VME will be adapted each phase of the SLM.

### Figure 3: Service Life Cycle

- Service identification: identify domain of the service, objectives, challenges for a transition according the Servitization process,
- Service concept: identify and define main concepts,
- Service requirement: identify, describe and model end-users required service,
- Service design: design, specify and simulate the provided service,
- Service delivery describes how the designed service will be delivery.

Between the various phases some feedback loops could happen, in order to answer better to the requirement of the previous phase.
### 3.3.2. From Virtual Manufacturing Enterprise to Manufacturing Service Ecosystem

The evolution toward Product2Service is certainly more complex and needs the cooperation of several types of enterprises and organizations in order to develop a strong potential of innovation. The reason is to answer to the strong competition on the market, therefore it is necessary to continuously adapt and improve the VME.

To enrich the potentiality of the VME, it is necessary to group different and heterogeneous entities like large OEMs, SMEs, Technical centers, Universities, research centers, individual professionals, employees, citizens and consumers etc. Such an organizational form is called in MSEE project, a Manufacturing Service Ecosystem (MSE). This MSE is left free to evolve and to network as it likes more, just following the market evolutionary law that it is the fittest species which survive. The MSE around supports and encourages this emergent and evolutionary approach by providing those entities with the necessary services round the product (Figure 5). Anyway it will be necessary to give a limit in the expansion and also some rules in order to maintain a coherence.

![Figure 5: Business Ecosystem Concept](image)

### 3.4. Service System and Service System Life cycle Management

The services in MSEE are produced by a Service System which could assume, as we have described previously different organizational forms such as VME or MSE.

#### 3.4.1. Service System

**Service science** aims to provide theory and practice around service innovation based on the notion of “joint value creation” among different roles offering and consuming services in a system.

In the description of the MSEE Servitization (3.1), we reach the conclusion that the creation of an Innovative Service is favored and in some cases requires a set of enterprises, research centers, customers etc. It is necessary starting from one manufacturing enterprise which sells one product to create a “System composed of various entities” that we call a MSE (Figure 5). Service system is considered to be the basic unit, in which entities perform actions to their mutual benefit. A service system consists of people and technologies that adaptively compute...
the knowledge about changing values in the system and adjust to it according (Chesborough, 2006).

To define a Service System in the domain of Manufacturing, we will use a comparison between the production of a product (Product System) and the production of a service (Service System) (Figure 6).

A **Product System** is composed of an organisation which produces a product and delivers a Product to the Customer. This organisation could be a set of enterprises, for example one being an OEM (Original Equipment Manufacturer) and the other being sub-contractors (i.e. Supply Chain). We call this organisational form a “Manufacturing Enterprise” and it is represented in the picture 6 by a rectangle with dotted line.

There are relations between the “Manufacturing Enterprise” and the customers but this type of relations is more knowledge exchanges and sometimes they are not strictly necessary. For instance, in the case of the manufacturing “on demand”, the collaboration between the Manufacturing enterprise and the customer is absolutely necessary and it is implemented by several exchanges of data, information and knowledge (e.g. about the product, its management, its usage, its maintenance, its disposal).

In a **Service System** which produces a Service, the customer is an integrated part: it is impossible to produce a service if the customer is not at the centre of the loop (even if in a manufacturing enterprise, the customer can be involved at the beginning of the loop) and data-information-knowledge is constantly shared between producer and consumer.

In a Service System, a very important phase is the realization/delivery of the service.

The service delivery system corresponds to the systematic and coherent organization of all the physical and human elements of the interface customer/enterprise necessary to the realization of service provisions whose commercial characteristics and quality levels of appreciation have been determined before (Boughnim, 2005) (Figure 7). The three elements required to deliver a service on a functional point of view are the customer, the contact people and the physical support.
3.4.2. Service System Life cycle Management (SLM)

As a basis, MSEE proposes to adopt the ISO 15704 (2000) standard which defines generic entity/system life cycle phases and evolution of the Service System in time. We have chosen this standard because it has a great influence in Enterprise Modelling languages and approaches that are used for each phase and is recognized by ISO. This standard has been developed by ISO TC184 SC5/WG1 (Modeling and architecture) on the basis of several enterprise architectures and methodologies (CIMOSA, PERA, GERAM, GRAI…). The main steps of this standard have been adapted to Service System lifecycle in the frame of the MSEE project:

- Service System identification: identify domain and existing component, objectives, challenges for a transition from product to service (or product + service)
- Service System concept: identify and define main concepts (models, functions, values,...) to create service around a product
- Service System requirement: identify, describe and model end-users required service system
- Service System design: design, specify and simulate the system that will provide that service
- Service System implementation: describe how the designed service system will be realized, delivered and implemented physically with all components,
- Service System operation: service system is operational for use by customers, this includes service consumption and interaction with customers, monitoring, evaluation, maintenance,…
- Service System decommission: end of the system service to remove and destruct it and recycle its components.

Figure 8 illustrates how an implemented Service System evolves in time (Service System lifecycle phases vs. Service System life). From Identification to Implementation, service system is designed and engineered following lifecycle phases. When a Service System is put...
in operation after implementation, it could be re-engineered several times during its life. Small changes might only need some redesign and implementation actions. Important changes might need to restart at concept phase to identify new/additional concepts and then to re-engineer part of or the whole service system following the lifecycle. At the end activities are also needed to disassemble and decommission the service system. A very important subject will be the link between the PLM and the Service System Life cycle Management particularly when we reach the phase of the reconfiguration of the product and related services.

Service life cycle

Figure 9. Class of Tools supporting Service Lifecycle Management

Figure 9 shows functionalities and purposes of tools to be provided by SP1 to support activities of all service lifecycle phases:

- **Services Identification and concept**: templates defined by WP 1.1
- **Requirement, Design and implementation**: Service Modelling toolbox specified by WP 1.1 and Service design and simulation specified by WP 1.2
- **Operation**: Monitoring and control Toolbox specified by WP 1.3.

Service KPIs, governance models and management tools will cover the SLM phases and will be specified in WP 1.3. The WP 1.4 will homogenize and integrate the various propositions for the SLM and WP 1.5 will develop the Toolbox.

Necessary service information captured and defined by various models will be stored in a repository so that a set of unified service data can be made available and used at any time for all stakeholders involved.

So, the structure of the VME could be different in the Design phase than in the Delivery phase and also of course the partners involved as shown in figure 10 below. For instance universities and research centres will be more involved in the ideation and design phase than in the realization phase.
3.4.3. Servitization and Service System evolution

With respect to the servitization four levels are identified to mark different stages in a servitization process of an enterprise: (1) Extended product, (2) Product + service, (3) Product to service, (4) Product as a service (please refer to Fig. 2).

According to this process, several transitions are possible even if one company could decide to reach only stage (2) or (3) and not mandatory stage (4):

- (1) to (2): Supporting services are identified, defined, realized, and they are offered together with the product: Ibarmia proposes the maintenance service,
- (2) to (3): A special kind of services are identified, defined, realized, and these services are recognized by the customer as benefits that differentiate the product from competitors: Ibarmia modifies the product by adding devices and sell a new service which is the continued checking of the machine in order to anticipate breakdown or failures, it means a service which guaranties a high availability of the machine. Such approach requires an environment in which the customer plays an important role with other stakeholder.
- (3) to (4): The product is rented to customers and remains the ownership of the enterprise, the service alone is to sell to the customer: Ibarmia gives the machine-tool and sells hours of machining.

We can imagine others transitions for example (1) to (3), (1) to (4), (2) to (4) etc. The process of evolution is quite complex.

In fact if we start from the stage (1), we could have a System enterprise which will be in an AS IS situation. Based on an innovation process, the enterprise decides to create a service and for that will move to a TO BE situation by cooperating with others System Enterprises (suppliers of services, manufacturers of devices,…..) in order to form a Virtual Enterprise (VE).

Depending on the market situation or others factors, the Virtual Enterprise which is in a new AS IS situation, will evolve by extending its activities (based on complex innovation process...
due to the need to become competitive on the market), requiring the cooperation with Technical centers, Research centers, consultants, plus the implication of other organization as Financial establishments. Such an evolved virtual enterprise requires a new situation (new TO BE) which is more easily achieved by a MSE.

The conclusion on the description of the servitization process is that the Modeling techniques for the Service System must allow to model a system enterprise but also a system of system, it means the VE or even, to a different detail, the MSE which generated it. In such a case, it will be necessary, by comparison of the AS IS model and the TO BE model at each stage, to determine the evolution of the Service System particularly:

- the IT system
- the Organisation and the Human which support this organization,
- Physical means as Machines, new devices or physical material.

In order to support the various transitions in the Servitization Process we propose a modeling support for the Service System with Concepts, Models, Tools and Methodology. This support must guide the evolution of the Service System in order to facilitate the determination of the components of the TO BE Service System starting from the AS IS situation.

### 3.5. The Modeling of the Service System

#### 3.5.1. Why to model the Service System?

We have already defined a Service System which could be a single manufacturing enterprise, a Virtual Manufacturing Enterprise (VME). In comparison, VME has precise objectives to reach in each phase of the SLM, although the Manufacturing Ecosystem is a group of companies without precise dedicated objectives to reach. For instance, a cluster of companies and research centres could be considered as an ecosystem but a particular composition of some members of this cluster in a dedicated project is a VME.

A Service System has the same structure than a Product System of a manufacturing enterprise, but is oriented towards the realization of a service. We need various functions as commercial, planning, accounting, strategy, etc… The difference is the production of Services combined with Products. We will analyse carefully this difference.

Based on our experience, we propose to be inspired for the Service System modeling by Enterprise Modelling concepts, models, methods and tools.

The advantage of this Enterprise Modelling approach is to be able to identify precisely the elements of the models (the concepts or the constructs of the model) using reference models and then to represent, to describe these concepts with adapted languages in order to deliver enterprise models. These enterprise models can be represented with several points of views: functions, decisions, business process, IT.

In several existing approaches, there is not this initial phase of creation of models and the starting is the use of languages which could generate misunderstandings and errors in the modelling.

Enterprise modelling techniques allows in particular:

- Facilitating the understanding of enterprise systems and improving communication and knowledge sharing between various stakeholders,
- Representing AS-IS (existing situation) and TO-BE (future situation) systems in terms of functions, business processes, physical system, decision system and IT system, and capturing business users requirements,
- Elaborating a diagnosis of AS IS i.e. strong points and points to improve, using specific rules and taking in account the strategy of the enterprise in terms of product and service proposition,
- Specifying the future system at various levels of abstraction through a model driven approach

The concept of system plays an important role in the Enterprise Modelling and by extension in Service System.

The most important lesson we have learned in our experience to apply Enterprise modelling in industry is the fact that it is necessary to consider two views: a global view which allows capturing the global structure, to understand the objectives and a local view which allows to model the detailed elements but in a coherent way with the global view.

Herbert Simon (March, J.G., Simon H.A), one of the founder of the System Theory, has written that “you can never know an enterprise or an organization, if you are not able to understand it as a whole but also if you are not able to represent the details and also to establish a link between the two views” (i.e. the global one and the local one).

Thus, the use of System theory allows to design, to understand and to represent the Service System.

However, it is necessary to describe and represent these models. So, it is necessary to use one or several languages depending of the nature of the concepts to represent and the point of view to represent. To define these languages we will use an approach inspired by Model Driven Approach recommended by OMG (Ref: Object Management Group) which is the leader for the development of languages. We will consider also ISO International standard (EN/ISO 19440:2006).

### 3.5.2. Modeling Service System using System Theory

We will introduce the System Theory and then the GRAI model which illustrates concretely the approach.

#### 3.5.2.1 System Theory

The system theory is the result of the research works done by many authors among whose we can refer to (H. Simon), (J.L. Le Moigne) and many others. The characteristics of these research works was that the same concepts (System theory concepts) were applied in various disciplines: biology, physics, economy, organisation, computer sciences, cybernetics.

From all these works, we can propose a reference model for Service System and several requirements for VME modelling.

A system is characterized by 5 properties.

1. A system is composed of a **limited set of elements** having attributes and relations between them, forming a particular structure. So the first question to model a system is:”What are the elements and the relations between them”. In the case of a Service System, it is necessary to identify the basic components as the products, the services, the manufacturing means, the IT resources, etc... It is recommended to start by a global description of these elements then according the needs to perform a detailed description.

2. These elements of a system and the structure contribute to reach one or several **common objectives**, in our case the objectives of the Service System. They could be economic objectives or technical objectives or social objectives. In MSEE, the objectives are related to the production of services based on manufactured product. The achievement of the objectives will be evaluated by performance indicators.
3. In order to reach these objectives **the structure of the elements must support several functions.** In MSEE, the functions can be related to the creation of services, the management of the resources, the purchasing of services or components, etc.

4. **A system has a boundary** which delimits the elements which belongs to the Service Systems and those which are outside. Sometime it is easier to determine the elements inside the system by determining the elements outside of the system. The elements outside the system compose the environment of the system and allow to define also the borders of the system. In MSEE it will be very important to determine the component belonging to the Ecosystem and the components outside in order to determine the behavior. This environment has the ability to modify the system properties. For example the market is the environment of the Enterprise System and it will influence the behavior of the Enterprise System.

5. **Finally a system is dynamic,** it means it evolves according to the time. The servitization process is typically a process of evolution. This capacity of evolution is the last property of a system.

A modification of one property of a system can lead to the modification of the various possible status of a system.

So, a system can be represented by the Figure 11 below:

![Figure 11. The structure of a system](image)

These concepts are a valuable reference for the modelling of one enterprise.

But in the frame of the servitization and the evolution toward a virtual enterprise or an Ecosystem, several enterprises or organizations must collaborate in order to form a complex Service system. This leads to the concept of system of systems.

This system of systems has the same properties than a single system, i.e. a structure of elements, functionalities, coherent objectives, an environment, and its own evolution.

The concept of system of systems could be represented in the Figure 12 below:
However, even if this is difficult to represent, the figure 12 must indicate that between the systems there is more than common objectives. Otherwise, the complete organisation is not a system. So, the system of system must integrate common functions, structure and evolution.

**Based on these definition, the system theory aims to represent (to model) the realities of a system, concretes or abstract, highlighting at the same time the global and the detailed representation of this system.**

In the domain of the System theory, the main application to Enterprise at the international level was developed by University on Bordeaux since 1980: the GRAI (Graph with Results and Activities Interrelated) model and GIM (GRAI Integrated Methodology). The detail description is given in annex 3. We will give in the following the main characteristics. The other modelling methods as IDEF, PERA, CIMOSA or Tool as ARIS don’t propose a reference model based on system theory. So it is possible to design the detail aspect but the global aspect is missing.

### 3.5.2.2 Service System modelling based on GRAI model

In MSEE, it is proposed to adapt the GRAI approach (GRAI model and GIM) for the modeling of Service System as VME.

The GRAI model gives the Conceptual description of an enterprise or a Service system. GIM allows to instantiate the GRAI model to a particular enterprise or an organization and to build the conceptual model of a particular enterprise or a Service System.

The first principle is the control of a system (equivalent to the control of an enterprise) (Figure 13: The control of a System):
• The controlled system (or the operative system): based on the information received from the customers and also the “material” (could be physical means or information) received from the suppliers, the controlled system transform the input in products or services to deliver to the customers,

• The control system plans, organize, manage the transformation according the objectives. It receives feedback information from the controlled system and delivers actions or adjustments. It received also information from the environment of the system

Control System / Controled System

The second principle is the decomposition of the control system in two sub-systems: the decision and the information sub-system (Figure 14).

Decision sub-system generates the right decisions in order to control the Physical System. The link between the Physical system and the decision is performed by the information system which receives the information from the systems inside the company (mainly the
Physical system) and from the environment (external information). The decision is the result of the following equation:

\[
\text{Decision} = \text{Information} + (\text{Objectives, Decision variables or action means, criteria to choose the Decision variables, constraints on the Decision variables}).
\]

Figure 15: the hierarchical or temporal decomposition of the Decision System

The third principle is the decomposition of the decision sub-system according to two criteria: temporal and functional (Figure 15).

The control of the Physical system is the result of decisions taken according to various functions (to manage sales, to manage design, to manage engineering, to manage manufacturing, to manage delivery…). The decisions have various natures: strategic at long term (define the objectives), tactical at medium term (define or plan the resources) and operational at short term (to perform the actions). This Grid is called the “Functional Grid”.

At the cross of a function and a level of decision we define a decision center.

The decomposition principles are based on the “Hierarchical System Theory” of (Mesarovic) which decentralizes the decision-making, keeping a coordination in order to reach the objectives of the enterprise. This decomposition is called “Hierarchical decomposition” or “Temporal decomposition” because each level is determined by a time horizon of decision.

There is another Grid called the “Control grid” which synchronizes the production of Products or Service and the availability of the Resources.

These sub-systems (Physical, Information, Decision) can be described in details by using various modeling process and information modeling: Extended Actigram for Physical, Grai Nets for Decision and Class Diagram for the Information.

The fact to describe explicitly the control system (Decision plus Information) determines the elements which allow reaching the objectives.

The principles of GRAI approach allow to represent and to study the system at the global level and at the detailed level (it is the case at the level of System of Systems and at the level of System).

More details on the GRAI model and GIM are given in annex 3.

So, based on the service life cycle and the considerations of modelling, it is obvious that the models obtained with modelling the VME built for each phase of the SLM will be different. This difference is shown in the figure 16 below:
3.5.3. Definition of the languages to describe and represent the Models

3.5.3.1 Principles

Among the various approaches used for the definition of the languages, we propose to use as a reference, the OMG 4 level architecture (Object Management Group) described on the Figure 17.

At level M0, we find the real world, it means all the concrete components which allow to build a particular enterprise or a particular Service System.

At level M1, we find the abstract model which describes conceptually all the components of the real world based. This representation will be based on concepts/constructs interconnected to represent processes, information, decision, resources, Performance Indicators, etc…. It is necessary to use several concepts to represent the real world and to determine relations between the concepts. In fact we use a language to elaborate the model of the reality. The relation between the two levels is called Abstraction because the representation at level M1 is a conceptual view of the reality: the models

At level M2, we find the basic concepts/constructs which allow to build the language.

At level M3, the basic components of concept/constructs are defined.
This sub-chapter deals with issues at the levels M3 and M2. It will define and provide modeling language(s) that will be used to create models to represent and specify service systems at the detailed level. The principles to define enterprise language constituents are discussed in the next section. The approach to follow to define enterprise modeling concepts and constructs is outlined in chapter 3. The set of concepts and constructs that form the proposed service system modeling language of MSEE is presented in chapters 4 to 6.

Notice: for the VME (more limited and with a clear purpose, the service in our case), we could use all the abstraction levels from M1 to M4. For MSE (very large and heterogeneous with no specific targets), we will limit to M1 for tangible and intangible assets (virtualisation = abstraction M1)

### 3.5.3.2 Modelling language concepts and constructs

At the level M2, (Figure 17) a set of modeling concepts and constructs will be identified and defined (see chapter 4). The following definitions are adopted:

- A concept is a generic ‘idea’ representing a particular interest of modeling. Examples of modeling concepts are: activity, process, decision, event, etc.
- A construct is an element used for modeling, which is defined from a concept and enhanced with a set of attributes. A construct has template and/or graphical representations.
- A modeling language consists in a set of constructs and the relationships between those constructs.

The approach adopted is as follows:

- Identify a list of main concepts of service system modeling capable of capturing required service system characteristics
- Identify and define relationship between the set of adopted modeling constructs (class diagram)
- Define a template per modeling construct to describe and characterize the modeling concept
- Map modeling constructs to existing modeling languages (or to suggest developing new ones)

At the level M1, we find the models described based on the languages chosen at the level M2.
3.6. Architecture for Service System Engineering

In today’s constantly changing market, Service System must be constantly adapted to the evolution of the market. The design and the implementation of Service System is not one shot and static activity anymore but dynamically evolves to meet customer needs. We call this process Service System Engineering.

To reduce effort and time in Service System reengineering, we propose to develop, based on the proposed modeling approach, a Model Driven Service system Engineering called Model Driven Service (system) Engineering Architecture.

3.6.1. Service system engineering – basic definitions

A service system varies from its most simple form (e.g. the maintenance system for machine tools) to more complex ones such as for example ‘an electric car renting system in Paris’, or the whole Apple ecosystem in which a system of systems interacts via value creations.

As explained previously, a service system is a collection of interrelated components that are organized for a service related purpose, i.e. to design, to produce, to manage and to deliver services to customers. In the context of product-based services in virtual enterprise, a service system consists of any combination of resources belonging to three domains: IT domain, Organisation/Human domain (including management and organisation), and Physical Means domain (including machine, robot and any other material handling devices).

In MSEE, Service System Engineering aims at designing and implementing Service Systems following a structured methodological approach, providing a set of concepts, modelling languages, models and methods.

It provides various representations of a service system at different levels of abstraction to support the design, production, management and delivery of services.

3.6.2. Model Driven Service Engineering Architecture (MDSEA)

An engineering architecture specifies a framework (i.e. a conceptual structure) for engineering activities, which provides a set of guidelines for structuring the specifications that are organised around various abstraction levels.

The objective of a model driven approach is to separate the different preoccupations from a business point of view of the product-service system to the technical preoccupations.

So, an engineering architecture specifies a framework (i.e. a conceptual structure) for engineering activities, which provides a set of guidelines for structuring the specifications that are organized around various abstraction levels.

The proposed Model Driven Service Engineering Architecture (MDSEA) is inspired from MDA/MDI (Model Driven Architecture/ Model Driven Interoperability) (MDA, 2008; MDI, 2010) to allow supporting the needs of modelling the three types of service system domain component (IT, Organisation/Human and Physical Means). It is therefore considered as an adaptation and an extension of MDA/MDI to the engineering context of product related services in virtual enterprise environment.

MDA gives the goals that must be followed at each modeling level but without mentioning how to model and with which language. The MDI approach is more detailed but more focused only on IT.

This is why we developed an extension based on these two model driven approaches

Similar to MDA/MDI, the proposed MDSEA defines a framework for service system modelling around three abstraction levels (Figure 18).

- Business Service Model (BSM): These models must represent the user point of view, understandable by the managers and business people. BSM level specifies the models, at the global level, describing the service running inside a single enterprise or inside a set of enterprises as well as the links between these enterprises. The models
at the BSM level must be independent to the future technologies that will be used for the various resources. In this sense, it’s useful, not only as an aid to understand a problem, but also it plays an important role in bridging the gap between domain experts and the development experts that will build the service system (adapted from Miller, et al., 2003). The BSM models allow also to define the link between the production of Products and the production of Services. In addition we state that:

- We develop the BSM in two sub-levels: the Top BSM and the Bottom BSM
- The Top BSM level is the result of the Modelling using the concepts of Systems, the models being described with the languages. It allows to analyse the options to develop the System Service,
- The Bottom BSM will allow to model in details the domain concerned by the servitization process using the same language than at the Top BSM,
- Based on this first analysis, the service system will be decomposed in the various domain components (IT, Organisation/Human and Physical Means) with a detailed description.

- **Technology Independent Model (TIM),** which delivers the models at a second level of abstraction independent from the technology used to implement the system. It gives detailed specifications of the structure and functionality of the service system that do not propose technological details. More concretely, it focuses on the operation details while hiding specific details of any particular technology in order to be suitable for use with several different technologies. At TIM level, the detailed specification of components of a service system will be elaborated with respect to IT, Organisation/Human and Physical means. The languages used at TIM level can be different to those at BSM but of course the models at BSM will be re-used to obtain the models at TIM level, by direct model transformation, but also with adding detailed information.

- **Technology Specific Model (TSM)** that combines the specifications in the TIM model with details that explain how the system uses a particular type of technology (such as for example IT applications, Machine technology or specific person). At TSM level, modeling and specifications must provide sufficient details to allow developing or buying software applications, components, recruiting human operators / managers or establishing internal training plans, buying and realizing machine devices, for supporting and delivering services in interaction with customers. For instance for IT component, a TSM adds to the TIM, technological details and implementation constructs that are available in a specific implementation platform, including middleware, operating systems and programming languages (e.g. Java, C++, EJB, CORBA, XML, Web Services, etc). As previously mentioned, the models at TSM level can be obtained by model transformation from TIM level but also with adding detailed information.
Based on the specifications given at TSM level, the next step consists in the realization and the implementation of the designed service system in terms of IT components (Applications and Services), Physical Means (machine or device components or material handling), and human resources and organization ensuring human related tasks/operations.

The proposed MDSEA aims at providing an integrated set of modelling languages at the various levels of abstraction to support service system design and implementation. A desired service system will be first specified and represented globally from a business user’s point of view at the lower level of the global modelling. Then detailed technical modelling and specification will allow to determine the three types of components (IT, Organisation/Human, Physical means) that are necessary to realize the service system. Finally implementation related descriptions and specifications will be given with sufficient details to allow procure these three types of components in order to build the design service system.

**3.6.3. MDSEA levels vs. Service system lifecycle phases**

Service system modeling under MDSEA will be performed following the system lifecycle phases. Consequently it is important to establish the relationship between the MDSEA modeling levels (BSM, TIM, TSM) and the system lifecycle phases (requirement, design and implementation). Error! Reference source not found. shows mapping of MDSEA three modeling levels to System Life Cycle phases.
As stated before, the modeling is a support to service system engineering activities following system lifecycle phases. Generally speaking, ‘Requirement’ phase consists in identifying the required service system from business user’s view. To support this, BSM modeling language is used to represent AS-IS and identify business user requirements of desired service system. The next phase is ‘Design’ which is generally split in user oriented design (also called conceptual design) and technical oriented design (also called detailed design). BSM language is used to model and describe user oriented design (user view of TO-BE) and TIM language is used to specify technical oriented design. Then TSM language will be used to further specify vendor dependent and technology specific system’s components (IT, Physical means, Organisation/Human) so that these specifications can be compared to market available offers in order to choose and buy (or develop if required components cannot be found in the market) those components. Finally at Implementation phase, BSM models are used to describe how those components are to be implemented (such as for example the layout of equipments and devices…).

- It is important to note that the design phase is concerned with the three modeling levels but each level has a particular focus. BSM model aims at representing business end-user requirements and specifications, and mainly deals with preliminary and end-user oriented design. TIM model supports technical and detail design, while TSM model focuses an implementation/realization oriented design.
- It is important to make the difference between the Hierarchical or Temporal decomposition (GRAI Model) and the Abstraction decomposition used in MDSEA. Indeed, the BSM models must cover the representation of the running of the system at the strategic, tactical and operational levels, from global and local points of view. At TIM and TSM levels the languages used are more focused on the operational level.

### 3.7. Conclusion

In this chapter we have given a set of definitions concerning Services, Service System and also proposed a modeling approach to take in account the global and the local modeling of a System Service. Based on this modeling we have defined an Architecture called Model Driven Service (system) Engineering Architecture (MDSEA) in order to generate the components to build the new Service Systems. For MDSEA, service modeling language concepts and constructs are defined at BSM, TIM and TSM levels (see chapter 4 to 6). This approach is derived from a long experience of the partners in the domain of Enterprise Modelling. A lot of applications in industry have been implemented in large company and also in SMEs.
4. SERVICE SYSTEM MODELING at BUSINESS SERVICE MODEL (BSM) level

In chapter 3 we have proposed basic concepts to perform Service System Modeling, particularly:

- the modeling of the Global Service System,
- combined with the modeling of the detailed Service System,
- the architecture called Modeling Driven Service Engineering Architecture (MDSEA) to enable to specify and if possible to generate the components of the Service System in the domains of IT, Organisation/Human and Physical Means.

In the chapter 4 we propose a set of core Business Service Modeling languages to model Service System at BSM level which is the first level of the MDSEA. Section 4.1 identifies relevant service system modeling concepts based on some existing enterprise modeling languages, methodologies. Selected concepts will be used as constructs to represent various aspects of a service system. Section 4.2 identifies relationships between adopted constructs. Section 4.3 defines a template for each construct. Section 4.4 maps existing relevant modeling languages to the set of constructs and proposes recommendations for the use of existing languages in MSEE project.

4.1. Basic service modeling concepts at BSM level

The goal of this first sub-chapter is to identify the basic concepts/construct to develop the modeling language at BSM level. In the following we give the main characteristics of some methods and language that are suitable for Service Modeling at BSM level. We recall that, at this level, we model the reality at an abstract level and the only way to check the validity of the model is given by the end-users. So the modeling must be understandable by non-IT specialist. The selection of the modeling techniques and languages has been done based on these criteria and also based on our knowledge after 20 years of work in the domain. We have also used the first input received with the requirements of the end users to elaborate a first check of the selected concepts and constructs.

4.1.1. Analysis of the identified Enterprise Modeling methods and languages

IDEF0 (Integrated DEFinition 0): developed by the US Airforce (USA) under the name of ICAM DEFinition (ICAM: Integrated Computer Aided Manufacturing, one of the most important project developed by US Airforce and one of the first important project to model Enterprise in manufacturing ). IDEF0 is derived from SADT (Structured Analysis Design Techniquess developed by Softech, a spin-off of MIT (Boston). It is a method with a language based on actigram representing the following concepts: activities, various flows, resources. The actigram can model mainly the concept of Service System Functions on a static point of view.

IDEF3 (Integrated DEFinition 3): Process Description Capture. Developed after the end of the ICAM project, IDEF3 proposes a method and a language to model Processes on a dynamic point of view (the main difference with IDEF0 which gives a static description). The language includes a Process Flow Description and Object State Transition Description. The IDEF3 language is used to describe processes and their behavior, similar to the use of Process Flowcharts and State Transition Diagrams. One important conclusion of the ICAM project in this time was: to represent an enterprise or an organization by consequence a System Service, it is necessary to model the function, the Information and the dynamic evolution by the simulation.
GRAI Integrated Methodology (GIM): a set of methods based on GRAI model developed by the GRAI Laboratory (Productic Group of IMS laboratory today) from Bordeaux 1 University since the eighties. The main originality of GIM is the GRAI model which allows the modeling of an Enterprise at Global level establishing with the local model and the Decisional Structure to control the running of the Enterprise. GRAI has reused several languages as Actigram (IDEF0), Extended Actigram, Class Diagramme (UML) for the Information modeling. Two new languages have been proposed with GRAI decision structure and GRAI nets. A tool has been developed by Graisoft: GraiTools now distributed by GFI and University Bordeaux1.

ECOGRAI: method based on GRAI model to define Performance Indicators with a specific language based on the concepts of GRAI decision.

ARIS BPM: from Software AG. ARIS is an integrated solution of several modules allowing to represent Business Process Model in an integrated way from corporate strategy to process design, from process design to automation, from automation to performance measurement. ARIS is certainly the most advanced industrial method, language and tool in the domain of BPM.

IEM: Integrated Enterprise Modeling. IEM is an enterprise modeling method used to model the Business Process in enterprises and in the public area and service providers. IEM is developed at the Fraunhofer Institute for Production Systems and Design Technology (German: IPK) Berlin, Germany.

The result of the analysis is given below in Table 1.

We have to take also in consideration three other inputs:

EN/ISO 19440: ISO 19440/2007 specifies the characteristics of the core constructs necessary for computer-supported modeling of enterprises conforming to ISO 19439. It focuses on, but is not restricted to, the computer integration of the information aspects of manufacturing, including the management and control technology and the required human tasks. It does not specify how these core constructs for model-based operations are to be implemented and, in particular, it does not include the control language needed to specify and execute (internal) activity.

POPO*: developed in the frame of the EC project ATHENA (Advanced Technology for interoperability of Heterogeneous Enterprise Networks and their Applications), an IP project from SP6, which has developed the meta models of the following concepts: Decision, Product, Process, Organisation, Infrastructure. The results are not yet exploited.

USDL: We have also analyzed USDL, the Unified Service Description Language (USDL) which proposed as a “master data model for services” to describe various types of services ranging from professional to electronic services. A detail description of USDL is given in chapter 7.

We give in the table 1, this list of Enterprise Modeling methods and languages and in the columns, we indicate if there is a method associated, a language, the main constructs, the type of Modeling Applications and the IT tool which supports the modeling.
<table>
<thead>
<tr>
<th>Name</th>
<th>Methods</th>
<th>Language</th>
<th>Main constructs</th>
<th>Modeling Applications</th>
<th>IT tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEF0</td>
<td>Method</td>
<td>Actigram</td>
<td>Activity Flows, Resources, Information</td>
<td>Functions model</td>
<td>Several Tools</td>
</tr>
<tr>
<td>IDEF3</td>
<td>Method</td>
<td>Process Modeling</td>
<td>Activity, Flow StateTransition Diagramme</td>
<td>Process Modeling</td>
<td>Several tools</td>
</tr>
<tr>
<td>GRAI Extended Actigrams</td>
<td>Method</td>
<td>Actigram Extended</td>
<td>Activity, Flow Resources</td>
<td>Process Modeling</td>
<td>GraiTools</td>
</tr>
<tr>
<td>GRAI</td>
<td>Method</td>
<td>GRAI GRID GRAI NETS</td>
<td>Activity, Flow Resources, Decision structure, Decision activities, Information, Decision frame, etc…</td>
<td>Global and local SSM</td>
<td>GraiTools</td>
</tr>
<tr>
<td>UML</td>
<td>Method</td>
<td>Class Diagram</td>
<td>Class Diagram</td>
<td>IS Modeling</td>
<td>Several tools</td>
</tr>
<tr>
<td>ECOGRAI</td>
<td>Method</td>
<td>Definition</td>
<td>PIs, Objectives, Decision Variables, Relations between O-DV-PI’s</td>
<td>Definition PIs</td>
<td></td>
</tr>
<tr>
<td>ARIS</td>
<td>Method</td>
<td>Process, Resource, Organisation, PI Management</td>
<td>A lot of constructs</td>
<td>Local SSM</td>
<td>ARIS Tool Set plus Web</td>
</tr>
<tr>
<td>IEM</td>
<td>Method</td>
<td>Process, Resource, control</td>
<td>A lot of constructs</td>
<td>Local SSM</td>
<td>MO2GO</td>
</tr>
</tbody>
</table>

Table 1. Identifying relevant Service modeling concepts at BSM level

4..1.2. Identification of the concepts for BSM level

The identification of the concepts for BSM level was based on the analysis of the Enterprise modeling list, the knowledge of the contributors, the review of the literature and also the analysis of the initial requirements supplied by the end-users of the MSEE project. The identified concepts for BSM level are:

- **Services**: defined in the context of servitization,
- **Products**: in such case there are the products which allow the servitization, the goal being not to define the product from a technical point of view but more from a management point of view, establishing the link with the production of services,
- **Customers**: involved in the definition of the service and all the SLM.
- **Partner**: a contributor to the SLM including the service,
- **Stakeholder**: all the entities involved in the SLM except the Customers and the partners (a research center for example),
Functions: defined in the frame of the Service System,
- Resource: Human, Physical mean, ITs,
- Business Process (BP): all the BP of the Service Systems,
- Organization: structures, responsibility, authority,
- Decision: decision structure of the Service System,
- Decision variable: action means to reach the objectives,
- Objectives: from the global objectives of the Service System (long term) to the detailed objectives (short term),
- Performance Indicators (PIs): from global PI (called KPIs) to the local PIs.

This list of concepts is considered as a list of core concepts, it means the main concepts to design the model of a Service System. Other additional concepts relevant to service modeling at BSM level can be further identified and added in the second phase of MSEE project.

4.2. Service modeling: relation between constructs at BSM level

The concepts identified in previous section are adopted as modeling constructs\(^3\) to represent a Service System at BSM level. Figure 20 defines the relationships between service modeling constructs. Each construct is further described by a template, a graphical representation and a text to explain some specific concerns and details.

![Figure 20: Modeling constructs and relationships at BSM level](image)

In MSEE, the considered service is related to a product. To develop such a service, it is necessary to build a Service System. Various entities are involved in such development from the Customer which will consume the service till the partners and the suppliers. Other entities (Technical centers, Research centers, bank, …) could be involved. All these entities are called Stakeholders and all of them may express their specific concerns. A Service is used / consumed by customers. A Service System provides functions which are utilities to fulfill customer’s needs. The Service provided can be evaluated by a set of Performance Indicators which are related to a set of decisions which control the Service System, i.e. are related to a set of objectives, and decision variables. The decisions are related in a decision structure the consistence / coherence of which must be analyzed.

The processes needed to provide a service to a customer must be defined and followed at run time. Generally speaking, a Service System is constituted by any combination of the three domains of resource (IT, Human, Physical means) which will be specified at TIM level, eventually at TSM. Finally, to ensure the good running of a Service System, an appropriate

\(^3\) Construct = Concept + its representation (template, graphics, text)
(hierarchical, decentralized) organization structure,…) will be defined. Details in terms of responsibility and authorization (who is responsible for and who is authorized to operate on what resource, process and decision-making) will be defined as well at TIM level and eventually at TSM level.

For instance, at the BSM level, the decisions are listed but not the related decision makers which will be defined at TIM level. Similarly, the domains of resources doing the activities are listed, but not the precise name or location of these resources.

Below is shown an example of one of our end-users case, INDESIT, modeled during one of the MSEE workshops. INDESIT plans to develop a carefree washing service. The case is globally described by the following templates:

<table>
<thead>
<tr>
<th>Service modeling construct</th>
<th>INDESIT Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Care free washing service</td>
</tr>
<tr>
<td>Product</td>
<td>W Machine (sold by Indesit)</td>
</tr>
<tr>
<td>Functionality</td>
<td>(1) remote control WM state, (2) flexible maintenance, (3) automate soap/cleaner recharge, (4) health &amp; safety control</td>
</tr>
<tr>
<td>Resource</td>
<td>Software for analyzing data feedback from WM, Sensors/Camera for house</td>
</tr>
<tr>
<td>Partner</td>
<td>Cleaning products provider, Local maintenance provider, Disposal and Recycling provider</td>
</tr>
<tr>
<td>Supplier/providers: (service providers)</td>
<td>Mobile application, H&amp;S system, local maintenance technique assistants, local home accidents assistants, software for maintenance</td>
</tr>
<tr>
<td>Other stakeholders</td>
<td>Energy suppliers, Water suppliers, Municipality, Engineer (Indesit)</td>
</tr>
<tr>
<td>Customer</td>
<td>WM owners</td>
</tr>
<tr>
<td>KPIs</td>
<td>WM Failure rate, number of failure avoided using new system, number of home damage accidents avoided, customer's maintenance cost, quantity of consumable products decreased, maintenance delay, n° of intervention, unavailability hours of WM, customer satisfaction, maintenance cost (Indesit), more ideas for design</td>
</tr>
<tr>
<td>Objectives:</td>
<td>Longer term relationship for loyalty, help to get more full life guarantee contract, Own confidence of customer to sell more machines, Make visible Indesit WM carefree ecosystem so that others can create additional ecosystems</td>
</tr>
<tr>
<td>Organization</td>
<td>(no possible to collect information)</td>
</tr>
<tr>
<td>Decisions</td>
<td>Scheduling maintenance actions (every week)</td>
</tr>
<tr>
<td>Process:</td>
<td>Alert intervention process, maintenance scheduling process, order Soap/powders process, spare parts supply process</td>
</tr>
</tbody>
</table>

Table 2: Example INDESIT Service Modeling constructs

Note: for each element in the template, a more detailed modeling can be made. The detail level will depend on the objective of the modeling. Figure 21 shows the graphical representation of the INDESIT case.
4.3. BSM service model templates

For each modeling construct defined in Figure 21, a template is also defined to describe in detail its attributes. The use of the whole or part of attributes is determined by each case and the objectives of the modeling.

4.3.1. Service

The service template aims at describing service information collected from end-users. More detail and specific information can be modeled at BSM level using some relevant modeling languages or tools such as Blueprint, UML use case diagram, etc. The service template is shown below.

<table>
<thead>
<tr>
<th>Header</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>Service</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the service instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the service instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Short textual description of this service instance</td>
</tr>
<tr>
<td>Objective</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationships to other model elements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT</td>
<td>Identifier/name of Product concerned by the service: described by Product template</td>
</tr>
<tr>
<td>FUNCTIONALITY</td>
<td>Identifier/name of functionality of the service: described by Functionality template</td>
</tr>
<tr>
<td>RESOURCE</td>
<td>Identifier/name of Resource providing the service: described by Resource template</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Identifier/name of Process providing the service: described by Process template</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>Identifier/name of Customer consuming the service: described by Customer template</td>
</tr>
<tr>
<td>PERFORMANCE INDICATOR</td>
<td>Identifier/name of the performance of the service: described by Performance template</td>
</tr>
</tbody>
</table>

![Figure 21: Links between INDESIT service case modeling concepts at BSM level](image-url)
### 4..3.2. Customer

The Customer template aims at describing generic characteristics of targeted customers consuming the service. Customer information will be used to better define the service and the system that provides the service.

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Identifier of the customer instance</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>name of the customer instance</td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>Categories</td>
<td>Worker, farmer, student,…</td>
</tr>
<tr>
<td></td>
<td>Level of education</td>
<td>Short textual description</td>
</tr>
<tr>
<td></td>
<td>Gender and age</td>
<td>Short textual description</td>
</tr>
<tr>
<td></td>
<td>Annual revenue range</td>
<td>Short textual description</td>
</tr>
<tr>
<td></td>
<td>Wish and requirement</td>
<td>Short textual description</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>SERVICE</td>
<td>Identifier/name of Service concerned by the customer: described by Service template</td>
</tr>
<tr>
<td>Other Relationships</td>
<td>RELATED TO MODEL LEVEL</td>
<td>Refer to BSM, TIM, TSM modeling level: BSM</td>
</tr>
<tr>
<td></td>
<td>RELATED TO SLM PHASE</td>
<td>Refer to service lifecycle phases: Requirement</td>
</tr>
</tbody>
</table>

Note: According to the specificities of a particular service, other additional customer attributes can be added in the template.

### 4..3.3. Stakeholder

A stakeholder is a person, group, or organization that has direct or indirect legitimate interest in a Service System. A stakeholder can be: a supplier, an employee of the enterprise, a local or national community, a retailer, an investor…

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Identifier of the stakeholder instance</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>name of the stakeholder instance</td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>Type</td>
<td>Textual description &lt;sponsor, engineer, provider, person in charge of maintenance,…&gt;</td>
</tr>
<tr>
<td></td>
<td>Concern</td>
<td>Short textual description of issues, interests, requirements,…</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
<tr>
<td></td>
<td>Role or position</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>SERVICE</td>
<td>Identifier/name of Service concerned by the stakeholder: described by Service template</td>
</tr>
<tr>
<td>Other Relationships</td>
<td>RELATED TO MODEL LEVEL</td>
<td>Refer to BSM, TIM, TSM modeling level: BSM</td>
</tr>
<tr>
<td></td>
<td>RELATED TO SLM PHASE</td>
<td>Refer to service lifecycle phases: Requirement</td>
</tr>
</tbody>
</table>

Note: Additional stakeholder attributes can be added according to specificities of a particular service.
4.3.4. Partner

A Partner is different from a stakeholder. A partner is a person or an organization associated with another in providing the service. A Partner is part of the Service System. A Partner has very strong relationships with the Service System in the MSEE. This is an “integrated stakeholder”.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>Partner’</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the partner instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the partner instance</td>
</tr>
<tr>
<td>Category</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Competence</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Concern</td>
<td>Short textual description of issues, wishes, requirements,…</td>
</tr>
<tr>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Contribution</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>Refer to BSM, TIM, TSM modeling level</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Identifier/name of service concerned by the partner: described by Service template</td>
</tr>
</tbody>
</table>

Table 6: Partner Template at BSM

4.3.5. Product

A Product is sold to a customer and is generally considered as a tangible physical entity. In MSEE, a product is related to one or a set of services in a virtual enterprise and ecosystem.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>Product’</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the product instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the product instance</td>
</tr>
<tr>
<td>Type</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Function</td>
<td>Short textual description of its utility or functionality]</td>
</tr>
<tr>
<td>Component</td>
<td>Short textual description; list of components, bill of materials]</td>
</tr>
<tr>
<td>Technical characteristics</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>Refer to BSM, TIM, TSM modeling level</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Identifier/name of service concerned by the product: described by Service template</td>
</tr>
</tbody>
</table>

Table 7: Product template at BSM

Note: In MSEE, a product will be modeled in detail at later stages to identify product use information that may need to support the service.

4.3.6. Functionality

A Functionality refers to any aspect of what a product or service can do for a user or customer. These are the reasons why a customer is interested in a product or a service, expressed by the customer. They do not include the technical functionalities, i.e. the technical way a functionality is performed, which are described at the lower levels (TIM and TSM). In the service domain, a functionality is a utility satisfying customer’s need(s). The functionality template aims at describing the functions provided by a service but not their technical characteristics.
4.3.7. Resource

A Resource is used and consumed in a service system to provide required services. A Resource is of three types: physical mean type (any kind of equipment); IT type (mainly software) and human type (operator, manager, administrator, knowledge...).

<table>
<thead>
<tr>
<th>Header</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>Resource</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the resource instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the resource instance</td>
</tr>
<tr>
<td>Type</td>
<td>Textual description &lt;human, physical, IT&gt;</td>
</tr>
<tr>
<td>Function</td>
<td>Short textual description: indicate role of the resource</td>
</tr>
<tr>
<td>Capability</td>
<td>Short textual description: missions realized by the resource</td>
</tr>
<tr>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>SERVICE</td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
</tr>
<tr>
<td></td>
<td>ORGANIZATION</td>
</tr>
<tr>
<td>Other Relationships</td>
<td>RELATED TO MODEL LEVEL</td>
</tr>
<tr>
<td></td>
<td>RELATED TO SLM PHASE</td>
</tr>
</tbody>
</table>

Table 9: Resource template at BSM

Note: At the BSM level and in some cases, the type of resource may not be fully determined. Only the function and required capabilities of the resource are expressed, to be then detailed at the TIM and TSM levels.
A Resource can be further modeled using POP* Product dimension constructs, EN/ISO 19440 or resource model of Smart Network methods. In case the physical Service System (composed of resources) needs to be described from the physical point of view, GRAI physical system modeling (based on Process modeling IDEF3) can be used.

### 4.3.8. Process

A Process defines a sequence of activities and may include sub-processes. In a Service System, various types of services processes, such as service booking process, service delivery process, service planning process, supply process, maintenance process, etc., are necessary.

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Identifier</td>
<td>Identifier of the process instance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
<td>Name of the process instance</td>
</tr>
</tbody>
</table>

| Body            | Short description of the process objectives |
| Trigger         | Condition this process instance is triggered |
| Result          | Output of this process instance |
| Constraint      | Constraint that may apply on this process instance |

| Sub-process     | List of sub-process [Graphical representation] |

Example: IDEF3 Process diagram

**Table 10: Process template at BSM**

Note: At the BSM level, a process can be further detailed and using various existing business-user oriented process modeling languages such as IDEF3, GRAI extended actigram, ARIS and IEM process languages.

### 4.3.9. Decision

A decision is the result of a choice among several alternatives to reach a certain objective under a set of constraints. It also means the activity of making a choice. In a Service System, the decisions are taken periodically or triggered by events. Those decisions control the physical service delivery process.

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>Identifier</td>
<td>Identifier of the decision instance</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
<td>Name of the decision instance</td>
</tr>
</tbody>
</table>
4.3.10. Decision structure

A decision structure represents the links between decision centers where decision activities are carried out. Using a set of defined GRAI rules, it is possible to analyze the consistence of a decision structure.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>Decision structure</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the decision structure instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the decision structure instance</td>
</tr>
<tr>
<td>Decision function</td>
<td>List of decision functions</td>
</tr>
<tr>
<td>Decision level</td>
<td>List of decision levels</td>
</tr>
<tr>
<td>Decision center</td>
<td>List of decision centers</td>
</tr>
</tbody>
</table>

| Relationship | Use GRAI grid to represent a decision structure |

<table>
<thead>
<tr>
<th>Relationships to other model elements</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECISION</td>
<td>Identifier/name of Decision concerned by the decision structure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Relationships</th>
<th>RELATED TO MODEL LEVEL</th>
<th>RELATED TO SLM PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refer to BSM, TIM, TSM Modeling level : BSM</td>
<td>Refer to service lifecycle phases : Requirement</td>
<td></td>
</tr>
</tbody>
</table>

Table 11: Decision template at BSM

Table 12: Decision structure template at BSM

4.3.11. Organisation

An Organisation refers to the way responsibility and authorization are defined and structured to ensure the good running of a system or enterprise. At BSM level, the organization...
modeling aims at capturing information relating to the type of (existing or targeted) organisation structure, such as for example hierarchical or decentralized. It is obvious that the organisation is linked to the modeling of business processes and of the decision structure. Indeed, a centralised or decentralised decision structure does not lead to the same organisation of human, physical and IT resources.

<table>
<thead>
<tr>
<th>Header</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>“Organization”</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the organization instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the organization instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Hierarchical, networked, decentralized, centralized,…</td>
</tr>
<tr>
<td>Description</td>
<td>Textual or graphical representation of the organization structure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationships to other model elements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCE</td>
<td>Identifier of resource concerned by organization: described by resource template</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Identifier of process concerned by the organization: described by process template</td>
</tr>
<tr>
<td>DECISION</td>
<td>Identifier of decision concerned by the organization: described by decision template</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Relationships</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATED TO MODEL LEVEL</td>
<td>Refer to BSM, TIM, TSM modeling level] : BSM</td>
</tr>
<tr>
<td>RELATED TO SLM PHASE</td>
<td>Refer to service lifecycle phases] : Requirement</td>
</tr>
</tbody>
</table>

Table 13: Organisation template at BSM

Note: At the BSM level, the degree of detail of the organization modeling depends on the needs of each specific case. To describe an existing organization structure, an Organization Chart can be used to represent the relationships between business units or people. There is a link between the organization model and the Decision GRID.

**4..3.12. Performance Indicators**

A performance indicator is a quantified data which measures the degree of achievement of a service in conformance with the objective and strategy of the company. To be efficient, a performance indicator must be related to a decision variable itself related to an objective. The performance indicator template is defined as below.

<table>
<thead>
<tr>
<th>Header</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>“Performance indicator”</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the performance indicator instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the performance indicator instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>customer oriented, provider oriented</td>
</tr>
<tr>
<td>Objective</td>
<td>Targeted objective by this performance indicator</td>
</tr>
<tr>
<td>Description</td>
<td>Short textual description of this performance indicator instance</td>
</tr>
<tr>
<td>Information required to build the indicator</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Processing required to calculate the indicator</td>
<td>Short textual description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationships to other model elements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Identifier/name of service concerned by this instance: described by Service template</td>
</tr>
<tr>
<td>DECISION</td>
<td>Identifier/name of decision concerned by this instance: described by Decision template</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Relationships</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATED TO MODEL LEVEL</td>
<td>Refer to BSM, TIM, TSM modeling level] : BSM</td>
</tr>
<tr>
<td>RELATED TO SLM PHASE</td>
<td>Refer to service lifecycle phases] : Requirement</td>
</tr>
</tbody>
</table>

Table 14: Performance Indicators at BSM
Note: To help identifying and defining a set of performance indicators to evaluate service and/or service system performances, the ECOGRAI method can be used. Other performance indicators might also be identified at the other stages of service lifecycle.

### 4.3.13. Value

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>&quot;Value&quot;</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the value instance</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the value instance</td>
</tr>
<tr>
<td>Description</td>
<td>Short textual description of this value instance</td>
</tr>
<tr>
<td>Relationships to other model elements</td>
<td>SERVICE [identifier/name of service concerned by this value instance: described by Service template]</td>
</tr>
<tr>
<td>Other Relationships</td>
<td>RELATED TO MODEL LEVEL [Refer to BSM, TIM, TSM modeling level] : BSM</td>
</tr>
<tr>
<td></td>
<td>RELATED TO SLM PHASE [Refer to service lifecycle phases] : Requirement</td>
</tr>
</tbody>
</table>

Table 15: Value template at BSM

### 4.4. BSM service modeling languages and tools: recommendations

The goal of this last sub-chapter is to give recommendations for the development of the tool that will support the Service System Modeling at BSM level. In the first part, we will analyze the potential tools which can support the modeling. Then we will discuss some alternative solutions.

#### 4.4.1. Potential IT tools to support Modeling at BSM

Service modeling constructs defined at BSM level can be supported by some existing enterprise modeling languages and IT tools. The IT tools can support several modeling languages and one language can support several constructs. We present hereafter the language and the IT tools which were considered for MSEE project synthesis, based on the initial study:

- **GRAI grid and GRAI nets** supported by GIM (methodology) and GraiTools (a Graphical language) allow the modeling of decisions and decision-making structure,
- **Extended Actigrams** allowing to model business processes
- **ECOGRAI** is a method for defining performance indicators. ECOGRAI is generally used with the GRAI grid modeling language. There is no specific tool yet. Business decisional tool can be used for the support of the management of PIs,
- **IDEF3** (Process description) is adapted for end-user oriented business process modeling. The Business Process of GraiTools is adapted from IDEF3, there are several tools which support the language of IDEF3,
- **BPMN** (Business Process Model and Notation) is an alternative solution to represent Business Process. BPMN is a graphical representation for specifying Business Process. It was developed by Business Process Management Initiative (BPMI). Now BPMN is maintained by the Object Management Group since the two organizations merged in 2005. The current version of BPMN is 2.0,
- **ARIS** modeling language supports the process modeling and Organization as well as Resource (human) modeling. ARIS is supported among others by ARIS EXPRESS tool,
- **IEM** (Integrated Enterprise Modeling) is another alternative for process modeling as well as resource and control (decision). IEM is supported by Mo2Go tool,
- **An Actigram** can be used to model functions and functionality and to decompose high level functions into sub-functions. There are several tools on the market,  
- **SmartNetwork** tool developed by DITF covers Process, Resource and Organization.

The Organization chart is adapted to represent the organization structure for describing both AS-IS and TO-BE systems.  
In the two pictures below, we have represented the modeling languages (Fig 20) and the IT modeling Tools (Fig 21) versus the BSM level modeling constructs.

![Diagram](image)

**Figure 22: BSM level modeling constructs vs. some existing modeling Language**

In Figure 22, we notice that several modeling languages are able to model the same construct, whereas several modeling languages are able to model several constructs.

However, based on the knowledge of the partners and on the coherence and complementarity of the languages related to system modelling, the MSEE partners have chosen the following languages:

- Extended Actigrams for process modelling. This language will be then improved to developed Extended Actigram Star
- GRAI Grid for decisional modelling
- UML for IT modelling
- ECOGRAI for PI’s

So, the figure 22 is then modified to obtained the figure 23:
Figure 23: BSM level modeling constructs for the chosen modeling Language

Figure 24 shows a mapping of existing IT tools mentioned above to support the BSM service modeling constructs. We notice that IT tools are able to support several modeling languages. Nevertheless, there are less IT tools than modeling languages and some constructs are not covered.

We notice that the notions of value, service, customer and stakeholder are not very well supported by the existing enterprise modeling techniques. Some USDL notations as well as service related literature found in the state-of-the-art are used as the basis to define the template attributes for describing those constructs at BSM level.

Generally speaking, for a given modeling purpose (for example process modeling), existing modeling languages and tools may use different semantics and syntax, focusing on different features and aspects of process representations.

Figure 24: BSM level modeling constructs vs. some existing IT tools

The Model information captured and collected using a particular language/tool can be mapped to templates. Figure 25 illustrates the use of process templates in order to compare the constructs used by various language/Tools.
4.4.2. Recommendations

We come back to the DOW:
“The first objective of WP 1.1 is to develop innovative concepts to describe Service Life Cycles in Manufacturing with models, methods and tools which will be used by the business users at their initial stage of a Service Lifecycle Management”.

We have explored the SLM and have defined a set of constructs to be able to model the Service System at BSM level.

“The second objective of WP 1.1 is to define a new modeling collaborative architecture, to support business-IT interaction and distributed decision making in virtual factories and enterprises. We will start from the architecture MDA/MDI that we have to adapt to service modeling.”

The MDSEA meets this objective. Nonetheless, WP 1.1 must deliver the specification of the tool to WP 1.5 after a phase of integration performed by WP 1.4:
“Main objective of WP 1.5 is to design and develop the software tools supporting SLM. These tools will combining heterogeneous services provided by the four other tools developed in WP 1.1 (Services Modeling), WP1.2 (Services Design & Simulation), WP1.3(Services Performance Assessment & Governance), WP1.4 (Service Life Cycle Mgmt in Virtual Factories & Enterprises)”.

We do not forget that these tools must support some phases of SP2, particularly the modeling of the Virtual Organisation and the Manufacturing Service Ecosystem. This implies the development of a flexible tool which will be able to be adapted along the life cycle of the project taking in account the two cycles.

One recommendation in the DOW is to not start from scratch. To cover all the constructs represented in Figure 23, we notice that GraiTools covers several constructs: Process diagram, actigram, GRAI grid, GRAI net, Class Diagram.
Nevertheless, the Process modeling has some weaknesses, although the following constructs are covered by other tools: resource, Performance Indicators, Service, Product, Value, Stakeholder, Partner and Customer. However, these constructs are not too complex to model, compared with Process modeling for example.

Moreover, we know from WP 1.2 that simulation must be specified and added. There will be a discussion if this tool will be integrated in the main tool or if it will be a complementary tool interfaced with the main tool.

The development will use the constructs defined in this deliverable.

4.5. Conclusion

We have proposed an initial set of modelling languages for this level. These languages are coherent in order to cover most of the identified constructs at the BSM level.

These languages are based on system theory in order to represent the VME at the global and local levels.

These languages are the base for the development of SLM TOOLBOX done in the WP1.5. These languages are applied to a specific pilot. The results of this application are proposed in the section 9 of this deliverable.
5. SERVICE SYSTEM MODELING: TECHNOLOGY INDEPENDENT MODEL LEVEL

The chapter 5 aims at presenting the principles of modeling at Technology Independent Model level to support the detailed design of service system in virtual enterprise. This chapter is structured in the similar way than chapter 4.
Section 5.1 identifies relevant service system modeling concepts based on some existing modeling languages, architectures, frameworks and methodologies. Selected concepts will be used as constructs to represent various aspects of a service system at TIM level.
Section 5.2 identifies relationships between the adopted constructs. Section 5.3 defines the templates for all the constructs. Section 5.4 maps existing modeling languages to the set of constructs and proposes recommendations for the use of existing languages inside MSEE project.
At this level we recommend to consider separately the three domains: IT, Organisation/Human and Physical Means. By this way, we retrieve the MDA/MDI architecture (CIM, PIM, PSM levels) for the IT domain.

5.1. Basic service modeling concepts at TIM level

We have to notice that there is a transformation model from the BSM level to the TIM level. In fact the Service System will be determined at the BSM level: often it is a part of the model elaborated at BSM level (see methodology at chapter 9).
The TIM level aims at detailing the Bottom BSM models concerned by the Service System that will respectively be supported by ICT, Organisation/Human and Physical means domains components. This level is not yet considering the use of any specific technology that will be done at the TSM level.
The research of service modeling at TIM level consists in identifying/defining necessary modeling constructs for designing a Service System at this level. In Table 16 (not exhaustive list), concepts from some existing relevant modeling languages, architectures, methods, generic or specific, are proposed to model the service system at TIM level.

<table>
<thead>
<tr>
<th>Existing approach</th>
<th>Existing relevant concepts</th>
<th>Service modeling constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>USDL</td>
<td>Service related concepts</td>
<td>interaction, access, consumption</td>
</tr>
<tr>
<td>WSDL</td>
<td>Service</td>
<td>Service</td>
</tr>
<tr>
<td>REST</td>
<td>Service, Resource, request, response</td>
<td>Resource (IT)</td>
</tr>
<tr>
<td>IDEF1</td>
<td>Data Entity, Relationship</td>
<td>Data Entity, Relationship</td>
</tr>
<tr>
<td>BPMN</td>
<td>Process, Functional Steps</td>
<td>Process, workflow</td>
</tr>
<tr>
<td>PERA</td>
<td>Human, Machine, Organisation</td>
<td>Human, Physical means</td>
</tr>
<tr>
<td>EN/ISO 19440</td>
<td>Responsibility, IT, human, machine resource, Organisation unit</td>
<td>Responsibility, IT resource, organisation unit</td>
</tr>
<tr>
<td>UML</td>
<td>Activity diagram, Class diagram</td>
<td>IT resource, Human resource, Physical means</td>
</tr>
</tbody>
</table>

Table 16: Identifying relevant Service modeling concepts at TIM level
5.2. Service modeling: relationships between constructs at TIM level

The concepts identified in section 5.1 are adopted as modeling constructs\(^4\) to represent Service System at TIM level. Figure 26 defines the relationships between those constructs. Each construct is further described by a template, a graphical representation and a text (if necessary) to explain some specific concerns and detail.

As defined in MDSE architecture, TIM level constructs can be divided in three parts to allow respectively specifying the services that will be supported by IT type components, Physical means type components and Human components (Figure 26).

![Diagram of Service Modeling constructs and relationships at TIM Level](image)

So, the IT related modeling part aims at detailing the model of the enterprise application, but mainly in terms of functionalities which will be implemented and without defining which application will be chosen. The functionalities will be of course derived from the models at the BSM level. These functionalities will be classified according to their importance in order to prepare the future selection at the lower level. The functionalities can also cover the requirements in terms of interoperability with other systems implemented already in the Virtual organization or the Ecosystem.

The Physical mean part will be related to the means to add value to the product and to support the service production. At this level, the functionalities of a specific machine and the expected performances can be proposed but no name of specific machine is proposed. As for the IT part, the functionalities can be prioritized and synthesized in order to propose a questionnaire that would be sent to the various machine providers.

Thus, the functionalities are derived from the models, in particular the models of the physical system and business processes.

The organization/human resource part aims at defining the type of organization and on human part the kinds of skills that are required according to the models. At this level, the name of a

\(^4\) Construct = Concept + its representation (template, graphics, text)
specific resource is not given but his/her place in the organization, his/her role in this organization and performance to reach must be detailed. Thus, this information can be derived, mainly from the decisional modeling and the business process modeling as well as from the physical models mainly for the persons directly involved in the product manufacturing and the service distribution. Moreover, these information will be used for the recruitment or the move of specific human resource as well as for the planning of the training of existing resource.

5.3. TIM service model templates

For the purpose of specifying service system components, some additional templates are created at TIM level. For the templates already defined at BSM level that need to be further refined at TIM level, they are detailed and extended. To keep consistency and traceability, attributes already defined at BSM level are kept in the templates.

As defined in Figure 18 (Model Driven Service Engineering Architecture), the modeling at TIM level aims at specifying the three types of resources (IT, Organisation/Human and Physical mean) that constitute a Service System. Consequently the resource construct is specialized in (1) enterprise application (including associated software, information and IT network), (2) physical mean (including any device and equipment dealing with physical objects) and (3) human (including organization issues).

5..3.1. Service

Service template defined at BSM level is further detailed at TIM level.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>&quot;Service&quot;</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier of the service instance</td>
</tr>
<tr>
<td>Name</td>
<td>name of the service instance</td>
</tr>
<tr>
<td>Description</td>
<td>short textual description of this service instance</td>
</tr>
<tr>
<td>Objective</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Constraint</td>
<td>Short textual description</td>
</tr>
<tr>
<td>Access</td>
<td>Specify how, where and when is a service made available</td>
</tr>
<tr>
<td>Consumption</td>
<td>Specify what is expected where and when by different parties participating in service</td>
</tr>
<tr>
<td>Automation level</td>
<td>Specify the decided automation / computerization level</td>
</tr>
</tbody>
</table>

Table 17: Service Template at TIM

Note: The part of service that needs to be computerized will also be further specified in enterprise application template at the BSM level.

5..3.2. Process

At the TIM level, process to be automated will be detailed and described as workflow.
Table 18: Process Template at TIM

Note: The part of process to be computerized will be further modeled in enterprise application template. The process that remains manually is defined at BSM level. If the BSM level description is not detailed enough, it can be refined and detailed at TIM level in order to expose clearly the use of any kind of resources (for which activity).

5.3.3. Human (resource)

A human resource can be an operator, a manager or any people performing an activity in the targeted service system. Human resource is derived from the Resource template defined at BSM level that requires a human to perform an activity.

At this level, the skill is more important, as well as the quantity and the expected performance.
Table 19: Human resource template at TIM

Note: ‘Other abilities’ refer to additional attributes such as the position to be occupied by this human instance, his/her role in the system etc.

### 5.3.4. Organization unit

An organization consists of organization units which can be a person, team, department.... An organization unit manages or controls a set of resources such as designers (design department), people in charge of maintenance (maintenance service), etc.

The organization will be mainly derived from the decision models and the business process models.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>[‘Organization unit’]</td>
</tr>
<tr>
<td>Identifier</td>
<td>[Identifier of the organization unit instance]</td>
</tr>
<tr>
<td>Name</td>
<td>[name of the organization unit instance]</td>
</tr>
</tbody>
</table>

| TIM Function | [Function of this organization unit] |
| Member | [List of members of this organization unit] |
| Responsibility | [Describe the responsibility of this organization unit] |
| Authorization | [Describe authorization of this organization unit] |

### 5.3.5. Organization

At TIM level the organization template defined at BOM level is further detailed. It defines what responsibility and authorization of each organization unit may have with respect to resources, processes and decision-making.

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
<td>[‘Organization’]</td>
</tr>
<tr>
<td>Identifier</td>
<td>[Identifier of the Organization instance]</td>
</tr>
<tr>
<td>Name</td>
<td>[name of the Organization instance]</td>
</tr>
</tbody>
</table>

| BSM Type | [Hierarchical, networked, decentralized, centralized,…] |
| Description | [Give a graphical representation of the organization structure] |
Physical mean type resource modeling aims at specifying machine, devices and equipment’s that are needed to provide and deliver product and related services. These components may include: robots, specialized machines and devices for producing, delivering, maintaining services, movers/transport means, IT equipments as well as any kind of physical facilities used for the creation and consumption of services.

**Table 21: Organisation template at TIM**

**5.3.6. Physical mean (resource)**

Information is used by Enterprise applications. Information and their relationships can be modeled using methods, such as Entity/ Relationship formalism, or IDEF1/IDEF1x language, or UML class diagram.
Table 23: Information template at TIM

Note: Information modeled and described at TIM level are for the purpose to further specify database(s) at the TSM level. Other types of information/data to be used or manipulated manually are not represented by this template.

5.3.8. Enterprise application

Enterprise application can include several kinds of modules with different functionalities as well as different databases. Functionalities identified at BOM level to support product and related services and supported by enterprise application will be detailed in this template. An EI instance can be for instance a module of the Enterprise Application or a software used (as a text editor for instance).
5.4. TIM service modeling language and tools and recommendation

Service modeling constructs defined at TIM level can also be supported by some existing modeling languages and tools. The following were identified of relevance:

- BPMN developed by OMG can be used to further model process information with more detail than IDEF3 (used at BOM level). BPMN is well adapted to describe processes (workflow) that will be automated (by enterprise application).
- EN/ISO 19440 is suitable to specify the three types of resource (IT, physical mean, Human) and to define organization unit.
- IDEF1 (IDEF1x) or E/R formalism is well adapted to model information and information relationships for the purpose to develop database
- WSDL is a XML format based language to describe service. It is usually used in conjunction with SOAP and HTTP (for web-based services).
- REST is similar to WSDL and is usually used with SOAP and HTTP to describe web-based services
- UML activity diagram can be used to model software based activities; UML class diagramme is similar to IDEF1 (E/R) and suitable to model information structure
- Responsibility matrix allows to specify responsibility of each organization unit with respect to processes, resources and decisions
- USDL notations provide concepts to further detail customer and service constructs as well as customer interaction
- SmartNetwork modeling developed by DITF can be used to model process, resource, service and organization as well.

Figure 27 shows the mapping of those existing modeling languages to service modeling constructs at TIM level.
**Recommendations**

The following recommendations are given:

- BPMN to model process that are to be transformed later to BPEL at TSM level for execution.
- UML class diagram to model information entities and their relationships for the purpose to build data bases
- WSDL to model service related applications that are to be provided by enterprise applications

So, the previous figure can be instantiated with the figure 28 below:

![Figure 27: Map existing languages/ tools to TIM level service modeling constructs](image)

![Figure 28: Map of chosen languages/ tools to TIM level service modeling constructs](image)
5.5. Conclusion

We have proposed an initial set of modelling languages for this level. We will have to check the validity of the concepts by applying in the use case of the MSEE project.

However because the models are not the same at this level, it is necessary to provide mechanisms to transform models at BSM level to models at TIM level.

So, it will be necessary to study for instance the transformation of Extended Actigram star into BPMN 2.0. A part of this work is presented in this deliverable but the complete work is presented in D11.3.
6. SERVICE SYSTEM MODELING: TECHNOLOGY SPECIFIC MODEL LEVEL

The chapter 6 aims at presenting the principles of modeling at Technology Specific Model level to support the design and implementation of Service System in virtual enterprise or Ecosystem.

This chapter is structured in the similar way with respect to chapters 4 and 5.
Section 6.1 identifies relevant service system modeling concepts based on some existing modeling languages, architectures, frameworks and methodologies. Selected concepts will be used as constructs to represent various aspects of a service system at TSM level.
Section 6.2 identifies relationships between the adopted constructs. Section 6.3 defines the templates. Section 6.4 maps existing relevant modeling languages to the set of constructs and proposes recommendations for the use of existing languages inside MSEE project.

6.1. Basic service modeling concepts at TSM level

Service modeling at TSM level aims at further specifying Service System components with sufficient implementation details related to a specific computing platform (for example JAVA) or vendor dependant technology (ex. SIEMENS). The service model at TSM level will allow:

- Coding software (including software-based service, process and all needed support functions), implementing related database and IT network,
- Choosing a specific legacy enterprise application based on the questionnaire developed at the TIM level,
- Choosing all needed commercially available physical means and devices from the market,
- Recruiting required personnel (both managers and operators) with adequate profile and skills, or define an internal training programme for existing personnel or a specific programme of mobility.

Most of TIM constructs are refined and detailed at TSM level with additional attributes or formalisms. Error! Reference source not found. shows relevant concepts from some existing modeling languages, models and methods potentially suitable for service system specifications at TSM level.

<table>
<thead>
<tr>
<th>Existing approach</th>
<th>Existing relevant concepts</th>
<th>Service modeling constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL, REST, USDL</td>
<td>Service</td>
<td>Service related concepts</td>
</tr>
<tr>
<td>BPEL</td>
<td>Process execution</td>
<td>Process execution</td>
</tr>
<tr>
<td>Database model</td>
<td>Internal schema, Physical schema</td>
<td>Internal and Physical schema</td>
</tr>
<tr>
<td>UML</td>
<td>Sequence diagram</td>
<td>Sequence diagram</td>
</tr>
<tr>
<td>ZACHMAN</td>
<td>Data structure, layout</td>
<td>Data structure, layout</td>
</tr>
<tr>
<td>PERA model concepts</td>
<td>IT, Machine, Human resources, automation level</td>
<td>IT, Physical mean functionality, Human resources skill, automation level</td>
</tr>
<tr>
<td>Other approaches</td>
<td>Layout, ServLab</td>
<td>Layout, ServLab</td>
</tr>
</tbody>
</table>

Table 25: Identifying relevant Service modeling concepts at TSM level
6.2. Service modeling: relationships between constructs at TSM level

The concepts identified in the previous section are adopted as modeling constructs to represent service system at TSM level. Figure 29 defines the relationships between service modeling constructs. Each construct is further described by a template, a graphical representation and a text (if necessary) to explain some specific concerns and details.

As defined in MDSE architecture, TSM level constructs are divided in three parts to allow respectively specifying the services that will be supported by IT domain components, Physical means domain components and Organization/Human components (Figure 29).

![Figure 29: Modeling constructs and relationships at TSM level](image)

So, the IT related modeling part aims at detailing the model of the enterprise application, but mainly in terms of detailed functionalities which will be bought or developed in the chosen enterprise application. The functionalities will be of course derived from the functionality modeling at the TIM level. The detailed functionalities can also cover the requirements in terms of interoperability with other systems implemented in one company of the ecosystem. The Physical means component domains will be related to the choice of a specific physical means. At this level, the functionalities of a specific machine and the expected performances will be verified and tested. As for the IT part, the functionalities will be very detailed and detailed actigrams could be used. This is obvious that the functionalities are derived from the models at the TIM level.

The human resource part aims at defining the specific resource which will be selected, to verify the appropriateness with his/her place in the organization, his/her role in this organization and performance to reach defined at the TIM level. This is obvious that this information can be derived, in coherence with the decisional modeling and the organization models at the TIM level.
6.3. **TSM service model templates**

Templates defined at TIM level are further detailed at TSM level in order to specify those characteristics that are needed to choose and implement service system components. As at TIM level, the three types of resources: (1) IT resources, (2) physical means (including any device and equipment dealing with physical objects and the layout) and (3) human (including organization issues) are refined with additional implementation oriented information.

**6.3.1. Process**

At the TSM level and for IT related part, process information collected and modeled at TIM level will be further detailed and transformed in a workflow description highlighting all the detailed information required to make the final choices of resources.

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>Name</th>
<th>Objective</th>
<th>Trigger</th>
<th>Result</th>
<th>Constraint</th>
<th>Sub-process</th>
<th>Execution</th>
<th>Other Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>‘Process’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>identifier of the process instance</td>
<td>Name</td>
<td>name of the process instance</td>
<td>BSM</td>
<td>Objectives</td>
<td>Short description of the process objective</td>
<td></td>
<td></td>
<td>RELATED TO SLM PHASE</td>
</tr>
<tr>
<td></td>
<td>TSM</td>
<td>Workflow</td>
<td>[Description of workflow]</td>
<td></td>
<td></td>
<td></td>
<td>Example: BPMN Workflow Diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TSM</td>
<td>Execution</td>
<td>[Description of process execution]</td>
<td></td>
<td></td>
<td></td>
<td>Example: BPEL process (NetBeans nifty graphical editor)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 26: Process template at TSM level**

**6.3.2. Organization unit**

Organization unit is TSM level refined to specify required implementation information in terms of responsibility and authorization assignment.
| Identifier | [Identifier of the organization unit instance] |
| Name | [name of the organization unit instance] |

**Body**

| TIM | Function | [Function of this organization unit] |
| - | Member | [List of members of this organization unit] |
| Responsibility | [Describe the responsibility of this organization unit] |
| Authorization | [Describe authorization of this organization unit] |

| TSM | Responsibility Assignment | [Assign responsibility to human instance] |
| - | Authorization Assignment | [Assign authorization to human instance] |
| Other implementation information | [Short textual description] |

**Other Relationships**

| RELATED TO MODEL LEVEL | Refer to BSM, TIM, TSM modeling level] : TSM |
| RELATED TO SLM PHASE | Refer to service lifecycle phases] : Design and Implementation |

Table 27: Organisation Unit template at TSM level

Note: At TSM level, responsibility and authorization as well as role are to be assigned to human resource instances (example: Paul is responsible to maintenance process, Jean is authorized to update service database, etc.)

### 6.3.3. Organization

At the TSM level and for the purpose of service system implementation, authorization matrix with respect to service system components is provided. It defines for each organization unit, the right, role and privilege with respect to resources, processes and decision-making.

<table>
<thead>
<tr>
<th>Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct label</td>
</tr>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

**Body**

| BSM | Type | Hierarchical, networked, decentralized, centralized,… |
| Description | Give a graphical representation of the organization structure |

| TIM | Responsibility relationships | Textual description and graphical representation |
| Authorization relationships | Textual description and graphical representation |

| TSM | Additional implementation descriptions | Description on where and how the organization is to be implemented |

Table 28: Organisation template at TSM level

### 6.3.4. Human (Resource)

Human resources will be further described as recruitment profiles at TSM level in order to recruit adequate people from work market or to define a training plan for internal personnel.
Table 29: Human (resources) template at TSM level

Note: Human resources attributes defined at TSM level can be accommodated in accordance with criteria and policies of Human Resource Department of a particular company.

6..3.5. Physical means (resource)

At TSM level, more detailed information need to be given in order to be able to choose (buy) machine type components from market including hardware. It is also necessary to define a physical layout in order to realize the physical part of the service system.

Table 30: Physical means (resources) template at TSM level

6..3.6. Layout

Layout describes the disposition and environment of physical system composed of physical mean type resources that deliver services. Simulation can be performed at this level to study various aspects of layout in relation to service delivery interaction with customer.
6.3.7. Information

Information entities and their relationships model defined at TIM level will be mapped to physical and internal information structure model in order to develop/choose and buy databases and enterprise application modules.

Table 31: Layout template at TSM level

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>'Information'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>[name of the information instance]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
</tr>
<tr>
<td>Data entities</td>
</tr>
<tr>
<td>Information model &lt;UML class diagram, Entity/relationship diagram,…&gt;</td>
</tr>
<tr>
<td>Relationships</td>
</tr>
</tbody>
</table>

| TSM    |
| Tables and indexes |
| Information schemas (internal and physical schemas) |

<table>
<thead>
<tr>
<th>Other Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATED TO MODEL LEVEL</td>
</tr>
<tr>
<td>RELATED TO SLM PHASE</td>
</tr>
</tbody>
</table>

Table 32: Layout at TSM level

6.3.8. Enterprise application

<table>
<thead>
<tr>
<th>Header</th>
<th>Construct label</th>
<th>'Enterprise application'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>[identifier of the EA instance]</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>[name of the EA instance]</td>
<td></td>
</tr>
</tbody>
</table>

| BSM   |
| Identifier | [identifier of the IT resource instance] |
### 6.4. TSM service modeling languages and tools

Service modeling constructs defined at TSM level can also be supported by some existing modeling languages and tools. The following non-exhaustive list shows some possible use of those languages and tools.

- **BPEL** can be used to further model process execution detail at TSM level. BPMN model used at TIM level can be transformed directly to BPEL.
- **WSDL** provides a machine-readable description of how the service can be called, what parameters it expects and what data structures it returns.
- **REST** like WSDL can also be used to specify service and IT resources with additional platform-specific details.
- **Internal and Physical schemas** are used at TSM level to specify further define information structure in order to realize/choose/buy database.
- **Ontology** technique is needed to describe data semantics. This will be based on adapted ontology practices and the ones developed along MSEE within the context of both Web 2.0 access and the coming longer-term vision of Semantic Web (3.0).
- **EN/ISO 19440** will provide concepts to define authorization (role, right and privilege) of organisation units with respect to process and resources.
- **Authorization matrix** presents a global view of defined authorization attributes to relate organisation units to process, resources and decisions.
- **UML sequence diagram** can be used to further model software based activities and relationships. UML can also be used to specify data tables and indexes.
- **USDL** provides additional service concepts to further specify implementation oriented service.
- **SmartNetwork of DITF** is also suitable to model those resources, in particular SmartService (DITF) contributes to specify ontology-enhanced service.
– Finally, ServLab as a 3D modeling tool can be used to model physical layout and to simulate human-resource interaction.

Figure 30 shows mapping of some existing languages to TSM level constructs.

**Figure 30: Map existing languages/tools to TSM modeling constructs**

**Recommendations**

We have given recommendation in chapter 4 for BSM level and for TIM level in chapter 5. The recommendation for this level must be coherent with the previous one.

At this level the specialist of the domains will be involved: IT, Organisation/Human and Physical means. The modeling techniques must be understood by these specialists, we don’t need to involve the end-user. They have been involved to validate the solutions proposed at the BSM level.

The following recommendations are given:

- BPEL to represent process that will be executed automatically in run time by enterprise application modules
- WSDL to further specify service related enterprise applications by adding specific implementation information
- UML to further model any other supporting enterprise applications that are needed to support service system
- Internal data schema and physical schema to further specify information elements and information structure for realizing/choosing/buying database.

**6.5. Conclusion**

We have proposed an initial set of modelling languages for this level. We will have to check the validity of the concepts by applying in the use case of the MSEE project.
7. USDL Service Data Repository

7.1. Service system models vs. USDL

Unified Service Description Language (USDL) developed by SAP aims at representing service data so that service can be put in the market for service commercialization, sale and delivery. USDL focuses on service data while Engineering models capture both service data and service system data. The common point of the two approaches is shown in Figure 31.

Figure 31: USDL vs. Engineering models

Figure 32 shows the difference of USDL and Engineering models (BSM, TIM, TSM). The complementarities in terms of role to play along the service system lifecycle phases are illustrated.

Figure 32: USDL support in Service lifecycle phases

Information collected and modeled in BSM, TIM and TSM models, relevant to service commercialization and delivery, will be projected to USDL repository to support service use
and operation. As engineering models can only provide part of data required by USDL, additional service data might need to be collected at USDL level.

### 7.2. Projections of SLM model concepts onto USDL model concepts

The modeling languages for SLM methods on BSM, TIM, and TSM level differ in their intended usage from USDL modeling language. Regarding the modeling constructs in these languages, the modeling constructs from the BSM/TIM/TSM models of the SLM approach are used to model the service system with respect to different modeling perspectives during all stages of the SLM service lifecycle.

For the level of models, the USDL models are perceived to be mostly relevant in the stage “Service Operations” of the SLM service lifecycle. The “Service Operations” deals with the phase when a service system is operational for use by customers including service consumption and interaction with customers, monitoring, evaluation, and maintenance.

With respect to USDL’s usage in the SLM lifecycle, this stage “Service Operation” may be subdivided into sub-stages as proposed here:

1. **Service description stage**: Service offer has been completed
2. **Service discovery and matchmaking stage**: Service offer has been published
3. **Service usage stage**: A service instance has been started, it is currently performed and the service experience is consumed by its service consumer(s).

Within MSEE the main purpose of USDL models is to model a set of selected service aspects that are relevant for sub stages 1 and 2, namely for activities like service description, service discovery, service composition or aggregation, and service consumption / service participant interaction. While some USDL model data can be used within sub-stage 3, this is not the primary focus of USDL support in MSEE (as of now).

Projection: The modeling concepts from BSM, TIM, and TSM model can be **projected** onto modeling concepts in the USDL modeling language (Figure 33). Modeling information from the BSM model, the TIM model, and the TSM model can be mapped into USDL models. Not all information from the BSM/TIM/TSM models is projected into USDL, and USDL can contain additional information not captured in the BSM/TIM/TSM models.
As an example, modeling information regarding participant information in a service system can be mapped from BSM/TIM/TSM models onto a dedicated USDL module “participants” that is used to model all service participant information in USDL.

Figure 33 gives an overview of the modeling constructs in the USDL module “participants”\(^5\). Organizations and persons can be modeled with a set of attributes (here: Organization, Person). Most prominently, roles can be modeled that are enacted by organizations and persons as their agents (here: Role, Agent). Within USDL, a set of pre-defined roles is modeled (Business Owner, Provider, Intermediary, Stakeholder, Consumer). Other modeling entities (here: NetworkedProvisionedEntity) can refer to such roles and can also refer to consumers explicitly (here: Target Consumers). The modeled relationships between these USDL modeling elements are also shown in the Figure 34.

---

With respect to participant-related information in the BSM models, TIM models, and TSM models, the following modeling constructs and relationships are relevant according to Sections 5-8 (not exhaustive):

- BSM model level: Customer, Resource, Stakeholder (and subclasses), Partner, Organization (and their relationships)
- TIM model level: Resources (and subclasses), Organization Responsibility
- TSM model level: Resource (and subclasses), Organization

These modeling constructs will be investigated in more detail and compared with the corresponding USDL modeling constructs in the USDL module “Participant”. The projection might not always be straightforward: For example, for each modeling entity, some model attributes may be different in both modeling worlds and the precise interpretation / semantics may be different in both modeling worlds. The final definition of the projection will be described in upcoming deliverables.

Similar to this example for a single aspect of the service modeling approach, all other relevant modeling perspectives of SLM will be investigated likewise and a selected subset of the modeling information will be projected from BSM/TIM/TSM models into USDL models.

For USDL, the following tools are available which all have been developed in the German research project THESEUS/TEXO:

- A USDL Editor to create USDL (XML) files can be downloaded at [http://usdleditor.sourceforge.net/](http://usdleditor.sourceforge.net/).
- A USDL file repository can be used to store USDL files as XML files. Users can be registered in the repository and USDL files can be stored and assigned to users. Services are provided through lightweight Web API. XML files that contain USDL model descriptions can be uploaded and downloaded into a repository as a whole (no incremental XML tree construction within the repository possible) and USDL identifiers can be used for retrieval of USDL files from the repository.
7.3. Conclusion

The service modeling language defined in this deliverable and the existing USDL are complementary. However, none of them can cover the modeling needs of the whole service system lifecycle phases. The proposed service modeling language aims at describing the service system and supports system engineering phases (requirement, design and implementation) while USDL is more oriented to the description of the service itself and supports more particularly the operation phase. Part of information captured during the engineering phases using the proposed service modeling language can be injected to a USDL database. The USDL database can be then seen as a service data repository to support service marketing, discovery, sale and delivery.
8. MODEL TRANSFORMATION METHOD

As introduced in chapter 4, MSEE (SP1/WP1.1) is using Model Driven Service Engineering Architecture (MDSEA) to contribute to the generation of the various domains components of the service system engineering activities. This is an opportunity to address many of the limitations of the traditional approaches to service engineering, by providing rigorous methods for capturing and integrating requirements, design, analysis, and verification along the SLM as well as facilitating the maintenance, assessment, and communication across the SLM. MDSEA is based on the MDA/MDI (explained in Annex 1) providing enhanced capabilities for service development as well as support to interoperability among services, offering more understanding between the developments teams in a Virtual Enterprise or Manufacturing Service Ecosystem.

The architecture proposed (i.e. MDSEA), unifies every step of the development of service system from its genesis, from a BSM the system’s business functionality and requirements level, through TIM and TSM, level to generate code or determine Application Software for IT and the specifications of the components domains for Organization/Human and Physical means in Virtual Enterprise or Manufacturing Services Ecosystem. Specifying services across an integrated three levels architecture reduces the risks of development since requirements validation and design verification can be done efficiently at a pre-implementation stage. This approach improve cost-effectiveness of the development process and also benefits from enhanced quality due more accurate traceability between requirements, design, analysis, and testing. Moreover, as an analogy with MDA/PIM and MDA/PSM, TIM models remain stable as technology evolves incrementing flexibility at TSM level while extending and thereby maximizing systems return on investment.

As in the MDA, portability and interoperability are built into the core of the MDSEA, introducing the distinction between vertical and horizontal transformations methods. The earlier implies a change on the abstraction level of the resulting model, thus going from BSM to TIM or from TIM to TSM implies a specialization transformation. In the case of horizontal transformations, which are used mostly to enhance interoperability (after the deployment of the services that did not start from the same top level models) among two or more organizations, the level of abstraction remains unchanged, leading to solutions for data integration within the modeling level to which the transformation applies. As illustrated in Figure 35, the both types of transformation (designated also by morphisms) apply to the three levels of the MDSEA. We know that the originality of the transformation is between BSM level and TIM level. The research work is very important due to the lack of efficient tools and methods.

In this deliverable, we will provide focus on the principles of horizontal and vertical transformations of the IT components at CIM (Computer Independent Model), PIM (Platform Independent Model and PSM (Platform Specific Model) levels.6

However, the deliverable D11.3 present further in detail the principles adapted to BSM, TIM and TSM as well as example of transformation between extended Actigrams Star and BPMN 2.0.

---

6 Due to the fact that in this part we consider only the IT development, the terms ‘CIM, PIM, PSM’ are kept instead of new proposed terms ‘BSM, TIM, TSM’.
8.1. Basic concepts and principles for MDA/MDI

Both input and output models considered in the MDSEA transformations must be an instance of a well-defined meta-model, and have to be classifiable according to the meta-modeling level they belong to (see section 3.6 “Architecture for Service System Engineering” for more detail). However, before entering into details it is important to clarify some basic concepts and principles for transformations:

The main objective behind model transformation consists in altering a source model A into a target model B, by means of modifying the first by a function $\mathcal{T}$ (Figure 36);

– Objects, service models, or domain ontologies, can be described as a model, e.g., in form of traditional Entity-Relationship (ER). As well, such models can be described by another higher level model, providing a metamodel dimension, corresponding to the topology associated to the model representation. This includes description of terminology (the labels of the model elements that don’t refer to modeling primitives), and also of semantics (the interpretations that can be associated with the model) (INTEROP NoE 2005),

– “Morphism” is a generic designation used throughout this chapter and in many of the related literature, to refer to model operations (e.g. mapping, merging, transformation, etc) (INTEROP NoE 2005). It can be seen as a function in set theory, the connection between domain and co-domain in category theory, or the edge that connects two nodes within a graph. Recently, this concept (MoMo) has been gaining momentum applied to computer science, namely to systems interoperability, as an attempt to formalize model based operations;

– The research community identifies non-altering and model altering morphisms as the two core classes of MoMo (see Table 28). In the first, morphisms are based on the
concept of traditional model-mappings, i.e., given two models (source A and target B), a mapping is created relating each element of the source with a correspondent element in the target (1-to-1, 1-to-n, or m-to-n), leaving both models intact. In model altering morphisms, the source model is transformed using a function that applies a mapping to the source model and outputs a different target model (Delgado et al. 2006). This function can be expressed by a simple table relating multiple or single constructs, but once it is created it can be later implemented by using more formal and executable languages (such as ATL\textsuperscript{7}, QVT\textsuperscript{8}, etc.). There are several techniques for achieving model transformations, at various levels, such as the top level “model-to-model” and “model-to-text” techniques (Czarnecki & Helsen 2006; Van Gorp 2008).

<table>
<thead>
<tr>
<th>MoMo</th>
<th>Formalization</th>
<th>Classification</th>
<th>Graphical Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping: $\theta(A, B)$</td>
<td>$\forall A, B \in M: \theta(A, B) \subseteq \text{Sub } A \times \text{Sub}(B)$</td>
<td>Non-altering</td>
<td><img src="image" alt="Graphical Representation of Non-altering Mapping" /></td>
</tr>
<tr>
<td>Transformation: $\tau: A \times \theta \rightarrow B$</td>
<td>$\forall A, B \in M: \text{if } \exists \theta(A, B) \text{ then } \tau A, \theta = B$</td>
<td>Model altering</td>
<td><img src="image" alt="Graphical Representation of Model Altering Transformation" /></td>
</tr>
</tbody>
</table>

Table 34: Classes of Model Morphisms

### 8.2. Vertical transformations

#### 8.2.1. Principles

Model Driven Engineering (MDE) uses MDA/MDI to specify a standard-based architecture for systems engineering, organized around three different abstraction levels that can be created based on vertical morphisms or transformations, for example specializations as in CIM to PIM or PIM to PSM and generalizations vice-versa. However, since the details of each modeling level are different, to perform such transformations is necessary to define a consistent set of rules that provide extra meaning to components from one abstraction level to another. This is normally accomplished thanks to a countless number of modeling languages that are already able to define rules for model transformation (e.g. UML).

The MDA/MDI methodology that has been incorporated into MDSEA can be applied to perform vertical transformations, using specific MDA/MDI tools to translate from one abstraction level to another and ultimately into executable code, which is in general anything textual, such as a XML document, a Java file or even text. Hence, to provide this transformation process with the required metadata and extra meaning to components, most of the MDA tools provide a mechanism to perform model annotations at the different abstraction levels (as illustrated in Figure 32). Additionally most of them have embedded the means to customize the transformation rules according to the user need, as well as pre-defined PSMs, with their respective annotation stereotypes for the most commonly used technologies and/or programming languages (e.g. Java, C++, C#).

\textsuperscript{7} ATL – Atlas Transformation Language (www.eclipse.org/m2m/atl/)
\textsuperscript{8} QVT – Query View Transformation (www.omg.org/spec/QVT)
Following the vertical process, the amount of generated code depends on both the code generator and also the level of detail represented in PSM (i.e. how well the PSM captures the details of the physical platform). Ideally, only small portions of missing code should have to be added by the human developer in order to ensure that the generated code and auxiliary files are ready for compilation, linking and deployment in the service ecosystem.

8.2.2. Tools and technology for vertical transformation

This section is focused on the existing tools for MDA/MDI which will be readjusted in the future to support the implementation of MDSEA. Some of the available vertical transformation tools, such as AndroMDA\(^9\), adheres to the MDE paradigm and implements the MDA methodology. Therefore, UML already provides a large support to PIM to PSM and PSM to code transformations. UML models can be represented both textually and graphically, and is also the core standard used to develop the Platform Independent Model (PIM) and Platform Specific Model (PSM). Moreover, besides its powerful modeling mechanisms, it has other features that are essential in an MDA environment, such as extension mechanisms – the UML Profiles. They are packaged stereotypes, which extend and annotate the UML language to accommodate new constraints, syntactic elements, or even to restrict it, thus enabling to add specific knowledge to existing models. This mechanism allows transformations to

\(^9\) AndroMDA – www.adromda.org
become more efficient using more complete mappings, or even to define a new language without the need of creating it from scratch (Fuentes-Fernández L, 2004).

Concerning MDSE, the same principle should be applied, and if models are formalized using the same technology, once the profile is well defined, and TIM or TSM models annotated, executable transformation can occur, achieving morphism automation from a model conforming to the defined stereotyped meta-model. However, today, MDI vertical transformations are still an open issue. Incomplete applications have been developed from CIM to PIM due to the lack of efficient tools and methods for transformation. For this purpose, new concepts, methods and tools are necessary. Among the confirmed enterprise modeling methodologies which can bring interesting concepts, we can highlight GRAI (Graph with Results and Activities Interrelated)\textsuperscript{10}, ‘House of ARIS’\textsuperscript{11}, or Leapfrog\textsuperscript{12}, but all need to be provided a suitable service engineering reference model to properly annotate BSM business models and enable vertical transformations to TIM before we can reuse MDA tools to achieve the desired TSM and deployable code.

### 8.2.3. Vertical Transformation Example

A vertical transformation example is illustrated on Figure 38, which represents the devolvement of a system based on a PIM model using the AndroMDA tool, which is a powerful open source extensible generator framework. It provides cartridges as UML profiles (containing stereotypes, tags and tagged values) so that any UML tool can be used to create the PIMs, and the profiles to mark them according to the cartridges specification.

The full example is available on Delgado (2008) and was originally meant to develop a package to create an industrial application to be applied in a collaborative platform for the aeronautics industry. Having the industrial application data model described as a UML PIM, and the UML model annotated according to the AndroMDA modeling patterns, the Java cartridge was used to generate Java classes from the model entities. Another AndroMDA cartridge is the spring cartridge, which is a feature based on the Spring Application Framework\textsuperscript{13}. It generates the methods signatures of services, letting the service behaviour to be manually programmed by the users. Additionally, AndroMDA provides a complete “ready to use” application, with dedicated web services to manage the entities in the PIM UML model.

\textsuperscript{11} http://www.ids-scheer.com/international/en
\textsuperscript{12} http://www.leapfrog-eu.org
In the case illustrated on Figure 38, the PIM model describes a formal means for defining an engineering context that links products, resources and applicable knowledge to perform a business activity within a project, and the application generated by AndroMDA is a set of dedicated web pages that permits browsing and search services for each manageable entity providing support to the collaboration environment used in the aeronautics industry. This example should be extended to accommodate MDSE.

8.3. Horizontal transformations

8.3.1. Principles for Horizontal transformation

Horizontal transformations are a current practice to enable interoperability among two organizations, each with their own service system. With them, it is possible to specify P2P mappings to translate any data provided from one side’s format specification into the other, thus allowing a seamless exchange of information.

When performing a horizontal model transformation (e.g. converting instances of a model to instances of another model) an explicit or an implicit mapping of the “meta-model” has to be performed. Thus, as depicted in Figure 39, the idea that when performing a transformation morphism at a certain level “n”, this transformation has (implicitly or explicitly) to be designed by taking into account mappings at level “n+1”. Once the “n+1” level mapping is complete, executable languages can be used to implement the transformation at “n” level, e.g. using ATL and the QVT (Eclipse foundation 2011; OMG 2008). This is valid either for MDI/MDA’s CIM, PIM, PSM, or MDSE’s BOM, TIM, TSM models. Although, interoperability between enterprises A and B is achieved, most of the times some information is lost, since the mappings might not cover all aspects of the mapped models. This can be a potential problem in the vice-versa transformation, therefore in order to guarantee interoperability it is needed to identify a minimal set of mappings.
This type of transformations are normally static processes that once defined can be repeated any number of times achieving the same results. However due to the constant knowledge change caused by the dynamics of the global market, services and models that regulate enterprise systems, are not. In fact, some researchers have attempted to extrapolate results from a “general systems theory” or “complexity theory” that could explain the importance of evolution of systems in all fields of science (Gharajedaghi 2007). These theories view all systems as dynamic, “living” entities that are goal-oriented and evolve over time, thus, information systems, services and the mappings that connect them should be prepared to respond to the environment dynamics which is in a constant update. To support this dynamicity, horizontal transformations should be provided with traceability features, and mappings stored in a parseable and structured knowledge-base (Agostinho et al. 2011).

### 8.3.2. Tools and technology for Horizontal transformation

The models mapping specifications can be performed either at a high level of formalization using graphs, sets, tuples, etc. or at lower levels, i.e. specifying the mappings by text. However, in both cases is necessary to implement them using a transformation language.

Model transformation is an important activity in MDE and OMG recognized this by issuing the Query/Views/Transformations (QVT) Request For Proposals (RFP) to seek an answer compatible with its MDA standard suite. Many contributions for the QVT RFP were submitted which led to several transformation languages with support for automatic model transformation execution. Some of these are based on the Object Constraint Language (OCL) (OMG 2010), like Atlas Transformation Language (ATL) and QVT itself.

QVT is a hybrid declarative and imperative transformation language that defines a standard way to transform source models into target models, which is sustained by the four levels of OMG’s meta-modeling architecture. It also supports bidirectional model to model (horizontal) transformations conforming to any MOF 2.0 meta-model. This means that model to text, whatever the text is (XML, Code, etc.), or vice-versa, is simply not supported. On the other hand, ATL is not so rigid and enables association with other methods to accomplish that. The main difference between them is that it can only be used to do unidirectional syntactic and semantic translation. An ATL transformation is composed by a set of rules (matched rules).
that define how the source model elements are linked, navigated enabling and instantiating the elements of the target model. These elements can then be filled with information from the source model by called rules (similar to functions in usual object languages like JAVA) and action blocks (blocks of imperative code which can be used by matched rules and called rules). ATL is one of the most used transformation languages, having a large user, nevertheless it is neither a standard nor a simple language to use (Bézivin et al., 2003).

The above languages are based on the OMG OCL, but others which are not can also be used and applied to horizontal transformations, e.g. Xtend/Xpand which is a JAVA looked-alike transformation language, and many others exist using similar paradigms, however they are not as well developed, e.g. UMLX, AToM3, MTL, just to enumerate some (Czarnecki & Helsen 2006). All the languages addressed in this section have some common goals and features, but also expose differences in their paradigms, constructs, underlying modeling approaches, etc. Despite the fact that they are designed as general-purpose model-to-model transformation languages, all have strong and weak points and demonstrate a better suitability for a certain set of problems. Comparisons of applicability and interoperability between several transformation languages are widely available (Jouault and Kurtev, 2007), which helps narrowing the choice to a few transformation languages for a known given type of transformation. Also, observing the executable languages and formalization techniques for managing morphisms (Agostinho, C. 2011), the conclusion is that some of them are ideal for structural issues, others for semantics providing good human traceability, while others are more formal and mathematical based. Although there are satisfactory solutions for transforming models to text (such as template-based approaches that enable PIM to PSM), this is not the case for transforming models to models (horizontal transformations) and even ATL which seems the most spread language, still lacks detailed documentation and a wider tool support. Nevertheless, from the analysed languages, ATL has the largest user base and a good JAVA integration.

### 8.3.3. Horizontal Transformation Example

This section presents a horizontal transformation example using services and MDI to achieve interoperability between two Enterprise Systems.

Figure 40 represents a way to integrate two companies that want to exchange data between them, using the same model language but different data formats. It is an instantiation of Figure 37, thus by applying a horizontal morphism at PIM level 1 and using ATL, automatic data transformation can be obtained at level 0. Both enterprises work with OWL models, so the objective is to have a mapping at information model level that can define how data can be transformed from enterprise A to enterprise B and exchange actual information throw the service bus at the bottom.
The mapping presented as an example relates the table “Contact” from enterprise A and the table “PersonalInformation” in enterprise B using knowledge enriched tuples as detailed in annex 2. In order to use ATL rules is necessary to execute some previous steps before applying the morphisms expressed in the tuples:

1. The best way to use ATL is to have the input models as an instance of XML meta-model which can be easily achieved since XML can be natively injected by the ATL modeling tools. Therefore, the first step in enterprise A side is to apply an XML injection to A’s model in order to obtain this model according to XML meta-model.

2. The second step is at level 2 executing a language mapping to represent the model according to the OWL meta-model.

3. Achieving this, is possible to elaborate the mapping tuples rules to map the two models from both enterprises and send that information to the service endpoint of enterprise B through the service bus.

4. To complete the data exchange, a similar process to the injection has to be done on the enterprise B side: first execute a language mapping at level 2 to obtain the model in XML meta-model and finish with an XML extraction to achieve enterprise B goal, i.e. the data according to B’s model.

Figure 41 illustrates the first mapping which links at conceptual level the two entities. It is not a perfect mapping since the table “Contact” has more expressiveness that the one used by enterprise B. To build the tuple we need a unique identifier (ID); the elements pair to map (Melems), which in this case are the concepts “Contact” (a) and “PersonalInformation” (b); the type of knowledge mapping (KMType) in this case conceptual; the classification for the match/mismatch – moreGeneral, stating that the concept from A is more comprehensive than the one from B; and finally the formal expression (Exp) needed to execute the mapping.

Figure 41: Tuple: A conceptual knowledge mapping

Figure 42 illustrates four remaining tuples that links at structural level the two tables. To note that tuple with ID 1.1.2.1 states that the table “Contact” from enterprise A contains fields that
will not be mapped to enterprise B side, more exactly the field “email”; the tuple with ID 1.1.3.1 represents a mapping of fields “Name” and “Surname” from “Contact” table to the field “Name” in table “PersonalInformation”; the tuple with ID 1.1.4.1 performs a perfect mapping between the field “Address” from both tables; and finally the tuple with ID 1.1.5.1. maps the field “Phone” from “Contact” table to “PhoneNumber” field in enterprise B table.

![Figure 42: Tuple: Structural knowledge mappings](image)

### 8.4. Integrated Transformation Example

This section presents an integrated transformation example using MDI and SOA techniques. Interoperability of 3rd generation (I3rdG), also called federal approach, uses ontology principles to adapt dynamically to the information exchange, and to bring more flexibility to an industrial workflow. This approach differs from the first and second generation of interoperability which respectively refers to the use of standard formats and meta-models to exchange information. Therefore, methodology proposed in the ISTA 3 (Interoperability of 3rd generation for Sub-Cotractors in Aviation Industry) project is based on MDI, supported by Enterprise Modeling Techniques, and Services Oriented Architecture (SOA), combined with Ontology to develop ISUs (Interoperability Services Utilities: services solutions adaptable and reusable). MDI permits the development of flexible interfaces between different information systems since these interfaces can be based on enterprise models, from the CIM level to the PSM level. The enterprise levels will permit the identification of the interoperability nodes from a process and decision point of view. Using SOA techniques is possible to define the technical services that will be supported and orchestrated by the service bus, and also provides numerous of services which are chosen based on the models derived from MDI approach. The main interest of the 3rd generation approach of interoperability is the combination of both techniques to bring automatic and then flexible generation of services based on the specific collaboration models.

The ISTA 3 methodology is composed of five phases (Figure 43) starts from business level (CIM Models) to PIM and PSM level to produce solutions using SOA techniques and Enterprise Service Bus (ESB) to orchestrate the services.
1. The first phase starts by evaluating the potentiality of the collaborations between the subcontractors. Before launching the collaboration, it is necessary to roughly evaluate the potentiality if each partner to work together. In this phase the method does not enter in detail of each system, it is performed a questionnaire on the knowledge of partners concerning their past collaborations.

2. The second phase determines the requirements according a CIM level modeling. This CIM level is decomposed in two levels: TOP CIM and BOTTOM CIM.

3. The third phase produces solutions using SOA techniques and ESB to orchestrate the services according the PIM and PSM modeling level.

4. The fourth phase implements the solutions inside the sub contracted enterprises. In this phase, the both partners IT systems will be modified in order to support the new interoperable services.

5. The fifth and last phase measures the interoperability performance in order to check if the initial objectives are reached and to maintain a continuous check.

Figure 44 illustrates a full integration between two previous examples, the horizontal transformation using tuples and the ISTA 3 methodology example. This combination is possible because both methods are based on MDI principles. Therefore, the three layers established in the first example can directly represent the CIM (TOP and BOTTOM), PIM and PSM levels or these layers can be applied to one precise level on the ISTA 3 methodology, for instance the CIM BOTTOM level. Consequently, is obtained a vertical transformation through the ISTA 3 methodology and a horizontal transformation sustained by the tuples. This example should be extended to accommodate MDSE.
8.5. Conclusions

In this deliverable, we developed on the problematic of the horizontal and vertical transformations of the IT components at MDA/MDI's CIM, PIM and PSM levels, looking forward for the future development of model transformations related with BSM, TIM and TSM that are presented in D11.3.

For the implementation of the basic concepts of MDA/MDI in the support of MDSEA, horizontal and vertical model transformation at each of MDA levels is core. Whilst the development of transformations between PIM and PSM is today a common practice, between CIM and PIM is not yet the case. Nevertheless, looking forward the implementation of MDI, the horizontal transformation between each of the MDA's levels is actually a major challenge that the research community is addressing now.

The proposal presented in this chapter is illustrated in D11.3 with a real example, providing a concrete proposal with clear indication how transformations can be implemented, and thus MDA/MDI can be effectively implemented.
9. METHOD

In order to operationalize the MSEE concepts and tools defined previously, it is necessary to propose a precise method to implement these concepts and tools.

This method will follow a structured multi-step approach and will be implemented through various groups of actors belonging to the enterprises and organizations involved in the servitization process as well as by external actors to support this process.

This project belongs to the FInES cluster (and also to FOF PPP). The FInES cluster has developed the principle of the future enterprise. Among the six main characteristics proposed (Inventive Enterprise, cloud Enterprise, Cognizant Enterprise, Community Oriented Enterprise, Green Enterprise and Glocal Enterprise) we select two of them: **Inventive enterprise** and **Glocal Enterprise** which seems very important for MSEE project. The reasons are:

- the employees must always think in term of innovation so they must know very well their environment and particularly the Service System,
- the need to understand the global structure of the enterprise and the link with the local activities (we refer to Simon, chapter 3).

For these reasons the method to develop the Service System must be participative to involve the employees. It is one of the reasons for which we have chosen GIM (GRAI Integrated Methodology) which implies strongly the end users.

![Figure 45: The various situations and the combination of companies](image)

If we analyse the transitions:

- From the situation 1 to the situation 2, the enterprise evolves on its own. So the approach will be more classical.
- From situation 2 to 3, the MSEE system will evolve from mono to multi enterprises,
- From situation 3 to 4, the MSEE system will evolve from a virtual enterprise to a Manufacturing Services Ecosystem.

9.1. Description of the structured approach

The Figure 46 below shows the proposed structured approach including all the required steps to go from the definition of the objectives of the transition to the detailed definition of all the required resources inside the enterprise and the Service System.
The horizontal sequence initializes the study to reach the TO BE for the Service System. The vertical sequence allows to implement the MDSEA in order to determine the components of the Service System by domains.
In the following, the various steps are detailed based on the adaptation of GIM (GRAI Integrated Methodology).

9.2. **Step 1: Definition of the strategy for the servitization transition**

The objective of this first step is to prepare the modeling of the Enterprise or the virtual enterprise by defining the strategy in terms of servitization. This step follows the innovation step that we have not indicated and which has determined the orientation to define the service.

Several methods can be used for step 1, as SWOT (Strength Weaknesses Opportunities and Threats) but also Brain Storming or Serious Games. The global objectives of the enterprise will be clarified in order to serve as a support and to derive the objectives of the servitization strategy. The result of this step is the strategy for the servitization transition. It will be interesting to anticipate some performances for the Service System.

9.3. **Step 2: AS IS Modeling at the BSM level, performance evaluation and diagnosis**

The objective of this step is to perform the modeling of the existing enterprise or virtual enterprise at the BSM level (AS IS model) and to determine the current performances. The
modeling allows to describe product, services, business process, decisions, IT, etc…(see chapters 3 to 6).

One of the difficulties of this step is to determine the domain of the enterprise to take in consideration (the boundaries of the System defined in chapter 3). Certainly we have to determine the part of the enterprise which is concerned by the servitization. The various models must be described giving complementary points of view as explained in the service system modeling (chapter 3): functional, physical, decisional, business process and IT, focusing on the various resources: human, technical.

The global models will be designed by involving the Synthesis Group (the main actors of the Service System) and the detailed models through interviews.

The approach followed during this step depends on the phase of the Servitization transition. It could imply one enterprise (case of the figure 40 with a transition from stage 1 to stage 2) or a virtual enterprise (VE). In such case, we will have to adapt the SG with the main actors of the VE.

![Diagram](image)

**Figure 47: The iterative approach of AS IS Modeling at the BSM level from phase 1 to 2**

The Figure 47 shows the various steps in the AS IS modeling approach and the iterations between the various groups.

The Project Board Group (PB) involves the main decision-makers of the existing enterprise (a maximum of 2 or 3 people). For instance, this group can involve the CEO, the service manager responsible for developing the new service and the industrial manager…(depending also on the size of the enterprise and on the part of the enterprise concerned).

Project Board’s missions are: to validate the objectives of the study, the intermediate results (various models) and the diagnosis on the present situation. During the first meeting PBi1, the Project Board has to initialize the study by:

- Analysing and validating the objectives of the study,
- Defining the global planning of the study,
- Defining the members of the synthesis group.

During the second meeting (PBi2), the Project Board validates the results of the AS IS modeling (the various models) and the orientation of the next phase (towards the TO BE) but also the proposed improvement based on a diagnostic.
The Synthesis Group is composed of the main persons responsible for the functions or the business processes concerned by the servitization. This group, composed of a maximum of 6 to 8 people (again, depending on the size of the enterprise) is in charge of elaborating the global models of the existing system with the help of one external or internal specialist of the method.

The Synthesis Group performs its work iteratively, based on the results of the interviews. When it is necessary to detail a part of the global model, Synthesis Group’s members can propose to involve the concerned manager of the Decision center or the process in order to get detailed information and to carry out the detailed model.

The detailed models are then validated by the members of the Synthesis Group.

The sequence to elaborate the models is very important. Generally, the modeling starts by a functional view in order to define the domain of the enterprise concerned by the servitization. Then is performed the modeling of the Physical System (called also Operative System, or controlled system) with the main technical resources and adding value activities on the product and services.

Once the controlled system is modeled, the modeling of the control system (Decision and Information systems) is performed.

This iterative modeling process by the Synthesis Group and the interviews allow after three or four iterations, to obtain the AS IS model which represents the existing situation with a strong degree of confidence. We must not forget that the modeling techniques must be understandable by the end-users who have been actively participating in the modeling process.

The Figure 48 below shows the structured approach in the case of transition from situation 2 to 3 or 3 to 4 because the modeling will involve not only one enterprise but a set of partners (suppliers, specialists, research organizations, etc…) to design the AS IS model.

![Sequence link](image)

**Figure 48: Iterative AS IS Modeling and Diagnosis at the BSM level from phase 2 to 4**

### 9.4. Step 3: Modeling of the TO BE for the MSEE (global and detailed) at TOP BSM level

Based on the objectives of Servitization and also of enterprise running improvement, this step aims at modeling the future Service System at the BSM level.
The approach is presented in Figure 49 below:

![Diagram of iterative approach of TO BE Modeling at the BSM level](image)

**Figure 49: The iterative approach of TO BE Modeling at the BSM level**

The “philosophy” of this approach is the same as the previous one but the interviews are replaced by the meetings of a Working Group, a group of enterprise specialists of one domain: for example Planning Scheduling or technology specialists, or marketing. The goal of the TO BE phase is to design the Service System model to produce the service defined. The involved people belong to the various partners of the virtual enterprise or the Manufacturing Service Ecosystem. These persons may be ERP specialists, specialists in IT systems, in service implementation, in marketing… So, the Synthesis Group will be in charge of elaborating the TO BE models based on the inputs of the Working Groups. This iterative approach has the advantage to ensure a coherent approach between the various Working Groups and to avoid any result which could be incompatible between the groups. The Synthesis Group ensures the coherence of the TO BE models at the BSM level.

9.5. **Step 4: TO BE at Bottom BSM Level**

Starting from the TO BE models at the TOP BSM (Business) level, this step is very important to ensure the coherence in the deployment of the models and to avoid non necessary works. This step consists in selecting the part of the service system that will be supported by new ITs, Human/organization or Physical means that will be developed at the lower levels. The main difficulty is to define the criteria that are used to select the BOTTOM BSM model. These criteria could be the accessibility/easiness of implementation, the potential of this implementation in terms of performance improvement… Of course, the results of discussion concerning the strategy are also very important. For this step, the modeling techniques used are the same as for the AS IS step in order to keep the coherence between the two level BSM.

9.6. **Step 5: TO BE at TIM Level**

At the TIM level, the previous models are refined using various modeling languages in order to represent more detailed concepts independently of the technology.
For the IT part we will use the MDA/MDI to transform the models from BSM level to TIM level. In term of concepts, as it was presented in chapter 5, the following recommendations are proposed:
- UML class diagram to model data entities and their relationships for the purpose to build data bases,
- WSDL to model service-related software applications
- BPMN to model processes that are to be transformed later to BPEL at TSM level.
Some of these models will be done based on the information gathered at the BSM level, but the detailed part of these models will be done through the collection of complementary detailed information.
We have recommended to build separately IT, Organization/Human and Physical means, knowing that the model at BSM level maintain the coherence.
For Organization/Human and for Physical means, a supplementary research must be performed to determine the results at TIM level.

9.7. Step 6: TOBE at the TSM level

At TSM level, the objective is to generate the components of Service System in various domains: IT, Organization/Human and Technical resources.

For the IT part, we will use the MDA/MDI to transform the models from TIM level to TSM level. In term of concepts, as it was presented in chapter 6, the following recommendations are proposed:
- BPEL to represent processes that will be executed automatically in run time by software applications,
- WSDL to further specify service-related software applications by adding specific implementation information,
- UML to further model any other supporting software applications that are needed to support the service system,
- Internal schema and physical schema to further specify data elements and data structure for realizing databases.
We have recommended to build separately IT, Organization/Human and Physical means knowing that the model at BSM level maintains the coherence.
For Organization/Human and for Physical means, a supplementary research must be performed to determine the results at TSM level.

9.8. Conclusion

We have presented the initial propositions for the method to implement Service Modeling. The reference is the GRAI Integrated Modeling which has led to interesting results in an industrial environment through several applications.
For MSEE, an important research work is still needed to implement the method, but we are confident in the potentiality of this proposal.
10. Application of the Modelling languages to a pilot case of MSEE

The goal of this chapter is to validate the concepts proposed in D11.1 and in D11.2 concerning the modeling of the service and the service system. The pilot case which was chosen is BIVOLINO.

10.1. Context of the company and servitization expectations

The objective of this paragraph is to summarise the context of the pilot case of BIVOLINO. We will describe the existing situation and in the servitization of Bivolino we will take only the first cycle it means the first evolution: evolving from the present situation (manufacturing shirts for Mark & Spencer to the Customized Cut file which could be sold as a service to M&S or other manufacturers of shirts.

At the origin BIVOLINO is a manufacturer of shirts. It is a family enterprise. The third generation is managing presently the enterprise.

From the origin, the enterprise has evolved starting from the manufacturing of shirts on demand (the customer) to the manufacturing in mass production with storage of shirts and externalization of the manufacturing in low-cost countries.

Now the Enterprise is coming back to the manufacturing of shirts on demand using the possibilities of the computerization of the design and manufacturing process to reduce the delivery time and about all to eliminate the storage.

The AS IS situation is to manufacture shirts for M&S, knowing that the customer of M&S use the Bivolino Configurator.

In such case, the various steps of the manufacturing are:

- The customer chooses the model of a shirt in a catalogue on a web site, indicating its choice for each component: collar, curves, fabrics, pocket shape with monogramme, shoulder shape, sleeves, plackets, buttons, base hem, contrasting,
- It is possible to order additional article as ankeshift, …., logo,
- Concerning the sizing, the customer has to give four numbers: height, collar size, weight, age
- The customer receive a confirmation of its order (Customer order),
- The Manufacturing Order (MO) is then sent to the manufacturing plant. The MO is accompanied by the Bill of Material (BOM) and the Cut file (file to control the cutting machine which will cut the components of the shirt in the plant). This phase produces the technical data for the optimization of the layout of the cutting of the fabrics including the “Alteration” (modification of the shape of the components to take in account some constraints),
- The cut file is loaded in the cutting machine: the production of the components of the shirt is launched,
- Then the assembling and the sewing is performed manually,
- The delivery of the shirt is performed at the end of the second day.

This description correspond to the AS IS situation. On figure 50 the AS IS situation must be considered without the “Interoperable cut file”.

Presently, Bivolino is manufacturing shirts for Mark & Spencer (M&S). It means that the customer of M&S is using the Bivolino system (under the label M&S) and the shirts are manufactured by Platico. For each usage of Bivolino System, M&S pays in addition the service.
Now we will consider the servitization of BIVOLINO case in MSEE project for the first cycle. It will be the TO BE situation: Development of a Customized configurator..

Based on several discussions with the specialist of M&S, the idea of the new service appears: Why not to produce the Cut File for the own production of M&S using a Bivolino customized configurator? In such situation M&Spencer can produce the shirts in its own factories. Bivolino has found the idea interesting and envisage to extend the service to other Manufacturers of shirts.

It is sure that the knowledge acquired by Bivolino since its creation particularly for the configurator with the “four numbers” and the Intelligent Data Base is an important immaterial asset and must be protected by Bivolino.

The main constraint will be the adaptation of the cut file to the various control of cutting machines existing on the market: Gerber Technology, Lectra System and ASSYST.

The cut file must be interoperable and also the language must be adapted to the control system (multi languages).

Let’s first distinguish the product ‘ configurator ‘ : the configurator is a frame in which consumers can define options for 3D visualization and ordering of a made-to-measure shirt: it’s based on an engineering database (fabrics, styleoptions, sizes, rules). Then we have the CC (configurator customizer) in order to be able to create configurable configurators for different customer/etailor/segments/targetgroups – for example a configurator for big&tal people (with so dedicated fabrics, size range, styleoptions) – this level 3 PRODUCT to SERVICE enables new product to be created/ordered (B2C). For example bivolino.com/h&m for uk obese men.
The components from the configurator are style options, fit, collars, cuffs, fabrics, contrast, monogram, sizes, extra’s, collection, prices.

The components from the CC are affiliate description: currency, country, metric sizing system, languages, payment systems + the configurator components (is in building/programming in MSEE).

Both are software.

10.2. Organisation of the information collection and modelling

The process was to collect the information from the main actors of BIVOLINO: Michel Bivoet and Gaby Ratajczak. The collection was done by Guy Doumeingts (INTEROP-VLab, Scientific Coordinator of MSEE) and Yves DUCQ (IMS, Univ. Bordeaux 1).

We have used partially the methodology described in chapter 9.

In fact the difficulty was to collect the right information to build the Service Product (SP) and the Service System (SyS) using GRAI model.

The Service Product is the product which allows to produce the Service. In the Bivollino case it is the Customized Configurator. We have to design, to develop and to exploit this Service Product.

The System Service is the system which allows to deliver the service to the customer including the contractual and the commercial aspects. So, this system must also be design and exploit.

One visit was organized in Bivolino (July 2012), two telephone conferences and a meeting in Roma on October 31st.

The approach is given in the figure 51 below:

![Process Diagram](image)

**Figure 51: The structured approach and models for the pilot case**

The interest of the approach is to perform the model in parallel to the evolution of ideas of BIVOLINO concerning the servitization.

The iterative process of modelling and information collection allows to obtain models which are progressively validated.
Then, the final objective is to differentiate the service product and the system design, in terms of process but also in terms of control of this process as presented in the figure 52 below.

![Diagram](image)

**Figure 52: The principles of process and control of service product and service system**

### 10.3. Presentation of the models

The models presented hereafter are done using the SLM TOOLBOX in order to demonstrate the coherence and the links between work done in WP1.1 and WP1.5.

The first model of the figure 53 below represents the global process of BIVOLINO with only one activity.

![Diagram](image)

**Figure 53: The global activity of BIVOLINO**
For that we receive the “Idea from the Customer”, we design, implement, operate till that the customer ends the cooperation Box.
This is represented by a “connector” for the Customer which sends a “flow” (“Idea of service”) in an activity which represents the transformation of the Idea of service in a service through a Service System. We represent also the end of the cooperation with the customer.

The decomposition of the global activity is presented in the figure 54 below:

**Figure 54: The decomposed activities of BIVOLINO**

Then, the model of the design activities is performed and presented in the figure 55 below:

**Figure 55: The design activities of BIVOLINO**

Then, the process model for the design of service product and of service system can be separate to be managed differently:
Figure 56: The separation of design activities of BIVOLINO

Then, the detail of the process for service product design is presented in the figure 57 below. We remind that the service product is the configurator customiser detailed in the figure 58 after:

Figure 57: The service product design activities process
The detail of the process for the service system design is presented in the figure 59 below:

Figure 59: The service system design activities process

Then it is necessary to detail the process of service system development as presented in the figure 60 below:
We can observe that the product and the related services are developed in parallel.

The last model that was performed is the model of the product service exploitation including the final customer, the customer (Marc and Spencer for instance), and BIVOLINO company. This model is presented in the figure 61 below:

Now we have built the process models for the Service Product and for the Service System.

So, we are presenting now the control system model of the Service product life cycle and of the service system, using GRAI Grid.

The GRAI grid in the figure 62 below aims at representing the control of the service product ideation, design and development:

![Diagram](image-url)
This GRAI Grid represents the three kinds of decisions: strategic with a horizon of 5 years and a period of 6 months, tactical with an horizon of 1 year and a period of 2 months and operational with an horizon of 2 months and a period of 1 week. This GRAI Grid shows also some interactions with the customer (as Marc and Spencer for instance) who order a new configuration of the configurator. The three functions cover the main phases of the service product life cycle.

Then, the decision model to control the service system exploitation is performed in order to help BIVOLINO to make the good decisions to optimise the service product use. This GRAI Grid is presented in the figure 63 below:

**Figure 63: The control system of service system exploitation using GRAI Grid**

We can observe that in this model, the decision levels are the same than previously but the final customer who buy a short play an important role in the decisions of the company. So, the decisions are then also at the strategic, tactical and operational level. The realisation and the delivery decisions are the same in the sense that both activities are difficult to separate.
11. CONCLUSIONS

The transition from a traditional product-centric manufacturing enterprise to a future service oriented virtual enterprise providing multiple services around a product is a complex project. From an engineering point of view, this transition is not a linear and straightforward process but needs multiple iterations at all stages of the design and implementation. Model based engineering approach is seen in MSEE as an important support to communicate and share information among stakeholders, determine design options, ensure traceability of all design decisions. It also facilitates reengineering by adaptation of existing models, thus reducing the effort and cost to make a service system evolve in the future.

Concretely we have introduced the modelling techniques to model the Service System putting in evidence the need to develop two kinds of modelling: the global modelling and the local modeling with the link between them. This approach is first recognized as essential to deliver a model close to the reality by the specialists of System Theory (H. Simon) but also corresponds to the recommendation of the FInES cluster. Our approach allow to take in account the following characteristic of the FInES enterprise: “GloCal” by the global and local model, “cognizant” by the Decisional aspects and “Inventive” by the interaction of all components of the Service System.

For the modeling languages, we proposed the model of OMG (Object Management Group) with four levels architecture (OMG 2000). We follow the recommendation of ISO 19440/2007 to specify core constructs of the language. The proposed languages will be developed based on of the GRAI languages that will be improved.

Service System engineering is no more one shot and static activity but dynamically evolves to meet customer needs. To reduce effort and time in Service System reengineering, we propose a Model Driven Service system Engineering called Model Driven Service Engineering Architecture (MDSEA). This MDSEA is decomposed in three level of abstraction:

- **Business Service Model (BSM)**, which specifies the models, on a user point of view at the global level,
- **Technology Independent Model (TIM)**, which delivers the models at a second level of abstraction independent from the technology used to implement the system.
- **Technology Specific Model (TSM)** that combines the specification in the TIM model with details that specify how the Service System uses a particular type of technology.

The present service system modeling language puts emphasize on the representation, at the BSM level, of Service System and service product. Service is owned and provided mainly by the company that owns the product, with the participation of multiple partners and providers, in close relation with customer who consumes the service. Such a loosely coupled virtual structure implies that participating companies are interoperable, at least for the parts that are involved in providing the service.

Service in its most complex form is provided by three types of resource: (1) IT resource for information processing, (2) Physical resources, (3) Organisation with human resource for manual operations. According to the desired degree of computerization and automation, TIM level modeling constructs will allow specifying these three types of resource, on the basis of desired conceptual TO BE model elaborated at BSM level.

Finally TSM level modeling will allow providing those specifications that can be mapped to market available information to purchase domains components. In the case of IT components, the use of MDA/MDI transformation model will allow to generate in a flexible manner the interoperable software.

USDL will be used as a complementary language to the Service System modeling language in order to support all service lifecycle phases from requirement definition down to implementation and operation.

A methodology to implement the modeling of Service System has been proposed to support the end-users.
This deliverable presents also, in comparison to D11.1 a real application of the service modelling concepts to a pilot case of MSEE.
In this application, the modelling enables to separate the service product and the service system design and exploitation, in terms of business processes but also in terms of decisions to control these processes.
The followed approach was top down with strongly involving the members of the company.

For the future works, first it is necessary to detail the propositions. It was not possible in this first phase due to the mount of works. Second the propositions must be disseminated in the MSEE project in order to inform all the partners but also to collect the feedback.
Anyway all the propositions need to be tested with the end-users of the project. Even if the team who developed the tools and the method has a long experience, only test in use cases will allow confirming the interest of the proposition and will bring a lot of feedback to improve the initial principles.
The first version of D11.1 is not complete: we will have to define more precisely the algorithms for the Model Transformation.
These improvements and new propositions will be presented in the Deliverable D11.2 at M12.
12. References


ATLAS Transformation Language homepage; Available at: http://www.eclipse.org/m2m/atl/; Last accessed on: 15th March 2010


Balin S. et Giard V. (2009) A process oriented approach to service concepts. 8ème Conférence Internationale de Génie Industriel. Tarbes, France


Berry, L., (1980), Services marketing is different, Business, 30 (May-June), pp 24-29.


Boughnim N., Yannou N. B., "Using Blueprinting for developing Product-Service Sytems", International Conference on Engineering Design ICED 05, Melbourne, 2005

Brezet, J. C. et al. (2001), The design of eco-efficient services; method, tools and review of the case study based ‘Designing eco-efficient services’ project. Ministry of VROM, Delft University of Technology.


Bo Edvardsson, Anders Gustafsson, Inger Roos, General review: Keyword(s): Services; Customers. International Journal of Service Industry Management Volume 16 Number 1 2005 pp. 107-121 Copyright © Emerald Group Publishing Limited ISSN 0956-4233


Christopher Lovelock, Jochen Wirtz and Denis Lapert (INT-Management, Evry, France)


MDI: Model Driven Interoperability; Available at: http://www.modelbased.net/mdi/; Accessed on: 15th March 2010


OMG, document number: formal/08-04-03.MOF 2.0 Query/View/Transformation 1.0.

OMG, Request for Proposal: UML 2.0 Infrastructure RFP. Available at [uml2wg], 2000


Orestis Terzidis, Alistair Barros, Andreas Friesen, Daniel Oberle, The Internet of Services and USDL, 2011, In Alistair Barros and Daniel Oberle, editors, Handbook of Service Description: USDL and its Methods, chapter 1,


Spohrer et al. “The Service System is the Basic Abstraction of Service Science “, 2008 41st Annual HICSS Conference Proceedings

Model Driven Development (MDD), also called Model Driven Engineering (MDE), represents a promising software engineering approach to address systems complexity, both by simplifying and formalising (standardising to allow automation) the various activities and tasks that comprise software life cycle (i.e. design, construction, deployment, operation, maintenance and modification). Given today’s increase of technology complexity, models are becoming more and more attractive as a powerful mechanism to precisely describe problems in a way that avoids delving into technological details, thus allowing a developer to focus on more abstract tasks and increasing productivity (Hausmann, et al., 2004).

In MDD’s vision, the primary artefacts are models, and their role in the development process should change from contemplative (e.g., used for documentation) to productive. This vision is driven by a unification principle, which states that “everything is a model” (i.e. programs, components, platforms, legacy software, services etc, are all models), similar to the basic principle in object technology stating that “everything is an object”, which was most helpful in driving technology towards simplicity, generality and power of integration (Bézivin, 2005). Besides the unification principle, another key principle in MDD is the support of models at different levels of abstraction, from high-level business models focusing on goals, roles and responsibilities down to detailed use-case and scenario models for business execution.

Among the several realisations of the MDD principles that exist, such as Agile Model Driven Development (Ambler, 2008), Domain-oriented Programming (Thomas, et al., 2003), Microsoft’s Software Factories (Greenfield, et al., 2004) and Model Driven Architecture (MDA) (MDA, 2008), MDA is perhaps the most prevalent at the moment. It is an OMG\(^\text{14}\) initiative, presenting a new vision of how application systems should be developed and managed, based on the MDD principles. Since it was launched, in 2001, it’s been having a major impact on the software development community, and presently, there is a large landscape of tools available for its support.

MDA specifies a standard-based architecture for models, which provides a set of guidelines for structuring the specifications that are organised around three different abstraction levels:

- **The Computation Independent Models (CIM)**, which specifies the requirements for the system and the environment where it will operate. It’s also called a domain model since it’s meant for the domain practitioners and it’s based on the vocabulary of the specific target domain. In this sense, it’s useful, not only as an aid to understand a problem, but also it plays an important role in bridging the gap between domain experts and the development experts that will build the system and/or services (Miller, et al., 2003).

- **Platform Independent Models (PIMs)** which are formal specification of the structure and functionality of the system that abstracts away technical details. More concretely, it focuses on the operation details while hiding specific details of any particular platform in order to be suitable for use with several different platforms. As an analogy, a PIM can be viewed as a model intended to be executed in a virtual machine (e.g. Java Virtual Machine - JVM) defined independently of any specific platform and which are realised in platform-specific ways on different platforms.

- **Platform Specific Models (TSMs)** that combines the specification in the PIM model with details that specify how the system uses a particular type of platform. For instance, in the MDA, the term platform is used to refer to technological and engineering details that are irrelevant to the fundamental functionality of a software component. Thus, a TSM adds to the PIM, technical details and implementation constructs that are available in a specific implementation platform, including

\(^{14}\) OMG – Object Management Group (www.omg.org/)

MSEE Consortium Dissemination: Public
Middleware, operating systems and programming languages (e.g. Java, C++, EJB, CORBA, XML, Web Services, etc).

Model Driven Interoperability (MDI) method is a model-driven method, based essentially on Model Driven Architecture (MDA) approach, to solve interoperability problems between enterprises not only at the application and software systems level, but also at the Enterprise Modeling level with an ontological support. This method aims at improving the enterprises performances, and it is supported through the extensive use of models in vertical and horizontal integration of the multiple abstraction levels defined in the Reference Model for MDI (Bourey J-P, et al. 2007, Berre A-J, et al. 2009). This method, as detailed on next Figure, introduces different conceptualization levels to reduce the gap between enterprise models and code level during the model transformation of MDD and Model Driven Architecture (MDA) sub-domains, and envisages the use of a common ontology to support both types of transformations.

![Figure 64: Reference Model for MDI](image)

The definition of the several levels in the Reference Model for MDI was based on the MDA, and as one can observe on the Figure 64, when compared to an MDA approach, the CIM level was divided into two sub-levels, i.e. the Top CIM Level (TCIM) and the Bottom CIM Level (BCIM) to reduce the gap between the CIM and PIM levels. This decomposition of the original CIM level lead to different characterizations for TCIM and BCIM:

- Top CIM is used to represent a company from the “holistic” point of view, i.e., its domain, business strategy, etc, on a high level of abstraction without any detail of the software applications features;
- Bottom CIM is the representation of the Top CIM, since it needs to be implemented on some computer system, but without linking it to any kind of technology or implementation in specific.

While the main objective of MDA is to separate the functional specifications of a system from the implementation details related to a specific platform, MDI method’s objective is to start at the highest level of abstraction and derive solutions from successive transformations, instead of solving the interoperability at the code level. Therefore, the interoperability model has been defined at the various different levels of abstractions, since this way can solve the horizontal interoperability problem between the enterprises.

Every mapping between models or ontologies of business partners can be stored and accessed by their local systems. This allows communities to build systems and services with reasoning capabilities able to understand each other’s representation format, without having to change their data and schema import or export processes. The methodology depicted in Figure 65: Method for Horizontal Transformations with Semantic Support, supports this need by defining morphisms in an MDA style with the aid of a knowledge enriched tuple that:

- Represents semantic mismatches among different model nodes;
- Specifies structural and semantic relationships to enable the reconciliation of divergent points of view of the same terms;
- And is stored on a communication mediator ontology for traceability purposes and providing a basis for sustainability/robustness support in a way that services can react (semi)automatically to model evolutions.

![Figure 65: Method for Horizontal Transformations with Semantic Support](image)

In more detail, it is possible to see that the method follows the MDA conventions of having data (level n) described by models (level n+1) which are defined by meta-models (level n+2). Thus, if one wants to enable automatic exchange of data, the necessary mappings between Enterprise “A” and Enterprise “B” need to be defined at the model level using the proposed knowledge enriched tuple (see next subsection for more detail). This tuple enables to specify relationships between different model elements and terms, as well as identify imperfect mappings, also known as semantic mismatches. Once this is done they can be stored in a mediator ontology and data transformations between both enterprises executed through a direct service bus connecting the services endpoints located on each one of the enterprises.

Knowledge Enriched Tuple for Mappings Representation

The research community has developed many proposals to morphisms representations (INTEROP NoE 2005), (MDA, 2008). As analysed in (Agostinho, C. 2011), graph theory has been used in some, although other theories can be considered to achieve the envisaged goals,
e.g., set theory (Dauben, 1979), model management (Bernstein 2003), or semantic matching (Sarraipa, J. 2010). However there is not a single perfect solution that can be used to achieve all the morphisms goals at once. Some are ideal for structural issues, others for semantics providing good human traceability, and others are more formal and mathematical based. For that reason, Agostinho et al. (2011) propose the usage of a 5-tuple mapping expression, with the objective to consolidate existent approaches to morphisms:

\[
\text{Mapping Tuple (MapT)}: \left< ID, MElems, KMTtype, MatchClass, Exp \right>
\]

Where, with M being a collection of models, LDMGraph a labelled oriented multigraph, and V the vertex set of a specific graph:

- \( \forall A, B \in M, \exists a \in A \text{ and } \exists b \in B \) If
  - \( M \text{ is an LDMGraph then } a \in V(A) \text{ and } b \in V(B) \forall A, B \in M, \exists a \in A \text{ and } \exists b \in B \)
  - \( M \text{ is an LDMGraph then } a \in V(A) \text{ and } b \in V(B) \)
    - \( ID \) is the unique identifier of the MapT and can be directly associated with the a’s vertex number
    - \( MElems \) is the pair \((a, b)\) that indicates the mapped elements
    - \( KMTtype \) stands for Knowledge Mapping Type (see figure below) and determines the type of mapping represented in a specific instance of MapT

  - \( KMTtype \) Conceptual, StructureSemantics, InstantiableData, Data

- \( KMTtype = \)
  - Conceptual, StructureSemantics, InstantiableData, Data
  - Conceptual, StructureSemantics, InstantiableData, Data
  - Conceptual, StructureSemantics, InstantiableData, Data

\[ \text{Figure 66: KMTType hierarchy} \]

- \( MatchClass \) stands for Match/Mismatch classification and depends on \( KMTtype \):
  - If \( a=b \), the mapping is absolute and \( MatchClass = \text{Equal} \);
  - If \( KMTtype = \text{Conceptual} \), the mapping is relating terms/concepts and therefore mismatches are seen as if relating sets (domains of knowledge) \( \text{Equal, Naming, Coverage, MoreGeneral, LessGeneral, Disjoint} \)
  - Otherwise the mapping is either non-existent or more concrete addressing structural issues. In this case \( MatchClass \) belongs to the semantic mismatches table of next subsection complemented with \( \text{Equal, Disjoint} \)
EXP stands or the mapping expression that translates and further specifies the previous tuple components. Normally, this expression can be translated to an executable transformation language, such as ATL (see section 9.3.1).

Example: \( a \geq b a \geq b \).

The idea of using a tuple brings many advantages, like being human traceable and readable or adding knowledge concerning mismatch. When used by intelligent systems, the tuple’s information enables automatic data transformations and exchange between two organizations working with different systems/services. According to the tuple philosophy, all the information about the mappings should be stored in a dedicated knowledge base (KB) so that it becomes computer processable, and morphism readjustments can be easier to manage and data exchange re-established automatically in a sustainable environment where services applications are capable of responding promptly to change of requirements and service descriptions. To reach these objectives (Sarraipa, J. 2010) proposed the Communication Mediator (CM) whose structure is explained in detail next.

### Semantic Mismatches

Mismatches are inconsistencies of information that result from “imperfect” mappings. Due to the differences among models, objects, concepts etc., almost in every case, a MoMo leads to a semantic mismatch, which can either be lossy or lossless depending on the nature of the related model elements (see Table 35): In lossless cases, the relating element can fully capture the semantics of the related; while in lossy mismatches a semantic preserving mapping to the reference model cannot be built.

This notion of mismatch can bring a semantic meaning to the type of the relationship being established in the mapping. However, the envisaged semantic “link” between two different models needs to account for more than inference of a meaning. It needs to be represented together with a formal expression (e.g. MapT) that is traceable and parseable by an intelligent service system which is able to deduce and recommend mapping readjustments whenever necessary (due to new requirements at the service level) and might even change the mismatch type (Agostinho et al. 2011).

<table>
<thead>
<tr>
<th>Mismatch</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming</td>
<td>Different labels for same concept of structure</td>
<td><img src="image" alt="Naming Example" /></td>
</tr>
<tr>
<td>Granularity</td>
<td>Same information decomposed in or composed by (sub)attributes</td>
<td><img src="image" alt="Granularity Example" /></td>
</tr>
<tr>
<td>Structuring</td>
<td>Different design structures for the same information</td>
<td><img src="image" alt="Structuring Example" /></td>
</tr>
<tr>
<td>SubClass-Attribute</td>
<td>An attribute, with a predefined value set represented by a subclass hierarchy (or vice-versa)</td>
<td><img src="image" alt="SubClass-Attribute Example" /></td>
</tr>
<tr>
<td>Schema-Instance</td>
<td>An attribute value in one model can be a part of the other’s model schema (or vice-versa)</td>
<td><img src="image" alt="Schema-Instance Example" /></td>
</tr>
</tbody>
</table>
Encoding | Different formats of data or units of measure
--- | ---
Content | Different content denoted by the same concept
Coverage | Absence of information
Lossy | 
Precision | Accuracy of information
Abstraction | Level of specialisation

Table 35: Semantic Mismatches (based on (INTEROP NoE 2005))

Communication Mediator KB

In aim to grant a planned traceability to support intelligence and sustainability, it was needed to store the to store the morphisms in a parseable and structured knowledge-base. With this, every mapping between mapping between systems or services can be accessed by their local systems. This allows communities to communities to build services with reasoning capabilities able to understand each other’s representation representation format, without having to change their data and schema import or export processes processes (Sarraipa, 2010). The proposed knowledge based (KB) is called Communication Mediator (CM) Mediator (CM) and is defined by ontology in OWL format, like illustrated in Figure 67: Structure of Communication Mediator KB

![Structure of Communication Mediator KB](image)

The knowledge is stored according to the tuple format, proposed in equation MapT and all the business partners in a collaborative network should have a CM in their local system, to act as a mediator for information exchange (Agostinho et al. 2011).

The structure of the CM, presented in the previous figure is described as follows: the CM has two main classes: “Object” and “Morphism”. The “Object” represents any “InformationModel” (IM) which is the model/ontology itself and “ModelElements” (also
belonging to the IM) that can either be classes, properties or instances. The “Morphism” associates a pair of “Objects” (related and relating), and classifies their relationship with a “MorphismType”, “KnowledgeMappingType” (if the morphism is a mapping), and “Match/Mismatch” class. The “Morphism” is also prepared to store transformation oriented “ExecutableCode” that will be written in the ATLAS Transformation Language and can be used by several organizations to automatically transform and exchange data with their business partners as envisaged before (Correia, F. 2010).

The next Figure 68 represents the five tuple instantiation (presented in section 8.3.3) using the Communication Mediator (CM) knowledge base.
Figure 68. Tuples instantiation

- **Table 1**: Tuple A
- **Table 2**: Tuple B
- **Table 3**: Tuple C
- **Table 4**: Tuple D
- **Table 5**: Tuple E
- **Table 6**: Tuple F

The diagram illustrates relationships between different tuples and their attributes, showing how data elements can be mapped or associated with each other within the MSEE (Manufacturing Services Ecosystem) framework.
15. ANNEX 3. Existing relevant service modeling languages

## 15.1. GRAI Model

### Name
GRAI conceptual Model

### Developer
University Bordeaux 1 – GRAI Laboratory and then LAPS and IMS Laboratory

### Objective
The GRAI conceptual reference model is a recursive structure which allows to represent with the same concepts, the global and the local models of a manufacturing system in an enterprise. In fact, the GRAI Model defines the various concepts that will be represented in the GRAI graphical formalisms. The interest of a conceptual model is to relate the various concepts in order to show their coherence, to avoid redundancies and to have a complete modeling.

### Background
The GRAI Conceptual model has been developed at the beginning of the GRAI Method development. It is continuously improved in order to define new concepts useful for GRAI modeling language. In particular, with the development of the different methodological modules (ECOGRAI, BENCHGRAI, GRAIPROGI…), the GRAI model has been improved.

### Description
The GRAI model is based on several theories: System Theory, Hierarchical System Theory, Organisation Theory, Discrete event systems, Production Management Concepts.

The previous concepts allow to consider the enterprise as a complex system which can be split up in a first time into two entities (figure below):

- the Physical system or controlled system (called Business system)
- the control system (or management system) which control the physical system.

The control system is very complex and it is necessary to decompose it in order to facilitate its modeling.

Two decomposition axis are defined:

- **A vertical axes linked to the nature of decisions**: The first criteria is a temporal one linked with the classical decomposition by level of decision: strategic, tactical, operational
  
  We assign to each level a temporal characteristic: the couple Horizon/Period:
  
  - Horizon: The interval of time over which the decision extends (i.e. remains valid).
  - Period: The interval of time after which we reconsider the set of decisions. In such a structure, the horizon is a sliding horizon.

  The couples Horizon / Period are linked. Inside a considered period of time, decisions can be made on event.
An horizontal axes linked to the nature of decisions: The second criteria of decomposition is the functional activities decomposition criteria. Generally, for typical enterprises, six functions are taken in account: to manage commercial, to manage design, to manage development, to manage production, to manage assembling, and to manage delivery. Other functions may be considered according to the type of enterprise. It is possible to develop reference models to bring new answer to this question. The interest of this model is to facilitate the integration between decisional levels and between functions. The matrix structure allows to co-ordinate the functional view (vertical) and the process view (horizontal) by decisional level. Moreover, there is a strong relationship between the physical system decomposition and the decisional levels, each level controlling a part more or less aggregated of the physical system. A decision centre is conceptually defined as the cross between a function and a decision level. The synchronisation of activities is performed horizontally, by decision level, the coordination being ensured through the decision levels. Each function of the enterprise can be decomposed more accurately taking in account the basic principles of the GRAI Model to manage the synchronisation of activities. At each decisional level, the performance objective imposes to synchronise in time the product and resource availability to perform the activity with a maximum of performance. For instance, considering the function "To manage production", we can build the grid to control this function with synchronising at the different decision level "The product management" activities, and "The resource management" activities. This synchronisation is obtained by the planning of the activities.

<table>
<thead>
<tr>
<th>Example</th>
<th>Supporting tool</th>
<th>Remarks</th>
<th>Open issue</th>
<th>References</th>
</tr>
</thead>
</table>
**15.2. GRAI modeling language**

<table>
<thead>
<tr>
<th>Name</th>
<th>GRAI modeling language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>University Bordeaux 1 – GRAI Laboratory and then LAPS and IMS Laboratory</td>
</tr>
</tbody>
</table>

**Objective**

The objective of GRAI Modeling language is to enable a complete modeling of the enterprise, from a structure and a running point of view. This modeling covers the functional, physical, decisional, IT, and business process points of view. The GRAI Modeling Language is based on the GRAI Conceptual Model describing the various concepts to represent, on modeling formalisms as GRAI extended actigrams, GRAI Grid and GRAI Nets and UML Class diagrams. Finally, GRAI modeling language is set up thanks to a complete structured approach describing the steps and actors involved in the modeling. The advantage of GRAI Modeling Language is to provide a set of coherent modeling formalisms allowing a global and a detailed modeling based on the system theory.

**Background**

GRAI Modeling language has been developed for more than thirty years starting from the decisional modeling in the 80’s and continuing by the complete modeling in the 90’s and the development of all the GRAI modules composing the GRAI Methodology in the 2000’s. These modules enable to use the enterprise models to go further in the enterprise engineering process: choose and implement an ERP, define a strategy, define and implement a performance indicator system… The first application was in the middle of the 80’s and then, supported by industrial and European projects, more than 200 applications have been done in industry and service companies, in big companies and in SME’s.

**Description**

THE GRAI Modeling Language is composed of several formalisms to describe the various parts of the company as described in the figure below:

- Actigrams for the functional view and physical system,
- Extended actigrams for business processes view
- GRAI Grid and GRAI Nets for the global and detailed modeling the decision system
- UML class diagrams for the IT system

Focus on the decision modeling:

The GRAI grid takes up the hierarchical and functional approach. It allows to identify the set of decision centres of the studied system, as well as their links. The GRAI grid is presented in the form of a matrix with :

- the managerial axis or control axis which represents the various decision levels which can be found in the enterprise. This axis is decomposed hierarchically in several levels, defined by the horizon and the period of decision making.
- the function axis which describes the various activities required to the product life cycle. It is decomposed into several functions which group a set of activities having a same identified finality (engineering, manufacturing, quality, maintenance, delivery, recycling ...).

Each function of this axis can be decomposed in three control functions: to manage the products (internal or external, it means procurement and purchasing), to manage the resources (human or technical) and to plan (to synchronize at each level product and resource management).

A decision centre is defined as the set of decisions made in one function and one decision level. The decision centres are controlling a part of the physical system more or less global depending on the decision level (more or less at a high level), the whole GRAI Grid controlling the whole physical system. The strategic decision centres control the whole company and the operational decision centres control a detailed part of the company: shops, production units...

The physical system is modelled with a top down approach using actigrams...
(IDEF0/SADT) formalisms. Objectives are defined for each decision centre as O_{PL,20}^2 for the decision centre at the cross of the function PL and the level 20 in the figure 1.

The example below was done in a service company specialized in financial engineering. The business process below is at the manager level. It aims at renegotiating the reimbursement of credit for the customer. This process is done in three steps and represented using extended actigrams:

The models below aim at representing the decisions to control the business process above at the global and detailed level. The GRAI grid presented describes the nine decision centres at the manager level:

The meaning of the various functions is described over the grid. This example shows the feasibility of the GRAI Modeling Language in the domain of services even if some adjustments could be proposed to better take into account the product at the same time than the service. The double arrows are the decision frames which represent the hierarchical links between decisions. The simple arrows represent information flows.

Supporting tool: The dedicated supporting tool is GRAITOOL. This tool, developed by GRAISOFT...
at the origin is now commercialized by GFI company.
This tool allows representing the various models of GRAI Modeling language: actigrams, extended actigrams, GRAI Grid and Nets, UML Class diagrams. The tool allows also the management of the study.

<table>
<thead>
<tr>
<th>Remarks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open issue</td>
<td>GRAITOOL is not yet an open source tool which leads to difficulties in the evolution of the tool.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
<th></th>
</tr>
</thead>
</table>
## 15.3. ECOGRAI

<table>
<thead>
<tr>
<th>Name</th>
<th>ECOGRAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>University Bordeaux 1 – GRAI Laboratory and then LAPS and IMS Laboratory</td>
</tr>
</tbody>
</table>

**Objective**

ECOGRAI is a method to design and to implement Performance Indicator Systems for industrial and service organizations. This method is applied with the implication of the decision makers of the organisation. There are two main steps in this method: design and implementation. The results of the design step is a coherent set of performance indicators covering the various business processes or functions and the various decision levels from the strategic to the operational. These indicators are materialised using specification sheets describing each Performances Indicator (indicators, concerned actors, required information and processings...). The implementation and the operating of the PIS are supported by a business intelligence tool.

**Background**

Created in the middle of the 90’s, at the same time that the well-known Balanced Score Card, ECOGRAI has been applied more than 100 times in various kinds of enterprises, in industry and services.

ECOGRAI has been developed through four main PhD of university Bordeaux 1, from its first concepts through various improvements concerning the coherence of performance indicators, the detailed steps for BI tools implementation, or for the development of a generic framework for PIS methods.

ECOGRAI is obviously based on the decision models keeping in mind that the definition of performance indicators is only useful if the decision makers can make real decisions based on performance.

**Description**

The main characteristics of the ECOGRAI method are:

- a logical process of analysis / design using a top-down approach, and allowing to decompose the objectives of the strategic levels into objectives for operational levels,
- a concrete process of participative implementation, creating a dialogue between the various levels of the hierarchy, and favouring the identification of indicators by the future users involved in the study: it is a bottom-up implementation,
- the use of a number of tools and graphic supports: GRAI grids, GRAI nets, splitting up diagrams, coherence panels, specification sheets,
- a coherent distribution of Performances Indicators covering the various functions and the various decision levels (strategic / tactical / operational),
- the search of a limited number of Performances Indicators by an original approach (figure 2). In a first step, we identify the objectives assigned to the decision makers (target situations which have to be reached inside the functions, depending on the decision level which is considered); in a second step, we identify the variables (called "Drivers") on which the decision makers can act to reach their objectives; and in a third step, we identify the Performances Indicators (quantified data which measure the efficiency of an activity or a set of activities of a function in the process to reach the objectives). In fact, the originality of the ECOGRAI method is not in the definition of Performances Indicators, but the search of Drivers on which decision makers can act to reach their objectives. The Performances Indicator is a consequence of the preliminary choice.
**The ECOGRAI original approach**

The logical structured approach of the method is decomposed into six phases (figure below). The first phase (phase 0) consists in modeling the control structure of the Production System and in determining inside the decision centers in which the Performances Indicators will be defined. The two following phases (phase 1 and 2) aim at identifying the basic elements which are required: the objectives and the drivers. The fourth phase (phase 3) consists in identifying the Performances Indicators, the fifth (phase 4) in designing the information system to build the Performances Indicators, and the sixth (phase 5) in implementing it inside the Production Management Information System.

**The six phases of the structured approach**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 0</td>
<td>Modeling of the control structure - GRAI grids and actigrams</td>
</tr>
<tr>
<td>Phase 1</td>
<td>Identification of the objectives and coherence analysis by performance aggregation</td>
</tr>
<tr>
<td>Phase 2</td>
<td>Identification of decision variables (DV) and analysis of conflicts between DV</td>
</tr>
<tr>
<td>Phase 3</td>
<td>Definition of performance indicators and internal coherence analysis</td>
</tr>
<tr>
<td>Phase 4</td>
<td>Design of the information system</td>
</tr>
<tr>
<td>Phase 5</td>
<td>Integration of the information system in enterprise information system using decisional tool</td>
</tr>
</tbody>
</table>

**Example**

Several examples were performed in particular in the domain of services in a financial engineering company.

**Supporting tool**

No supporting tool

**Remarks**

**Open issue**

**Reference**

M. BITTON - ECOGRAI : Méthode de conception et d’implantation de systèmes de mesure de performance pour organisations industrielles - Thèse de doctorat en Automatique - Université Bordeaux I - Septembre 1990

Y. DUCQ - Contribution à une méthode d’analyse de la cohérence des systèmes de Production dans le cadre du modèle GRAI - Thèse de doctorat de l’université de Bordeaux I – 1999

VALLESPIR B., DUCQ Y., DOUMEINGTS G.


DUCQ Y., VALLESPIR B.

15.4. POP*

<table>
<thead>
<tr>
<th>Name</th>
<th>POP*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>ATHENA Integrated project</td>
</tr>
</tbody>
</table>

**Objective**

POP* is an acronym for “Process, Organisation, Product etc”. The objectives are as follows:
1. The POP* meta-model, defining the concepts and their interrelations of the POP* modeling constructs;
2. A set of guidelines and scenarios describing how the POP* meta-model should be managed and used.

**Background**

POP* was developed under ATHENA Integrated Project (Sub-project A1 led by Thomas Knote from IPK Berlin) to develop interoperability of enterprise models. It can be also considered as a continuing effort of UEML (Unified Enterprise Modeling Language) European Thematic Network initiative under FP5. Main modeling concepts come from some existing enterprise modeling languages and methodologies such as BPDM, UEML, METIS, etc. Experiences from other publicly available standards (STEP, SCOR and HLA) are also used as a basis.

**Description**

The POP* meta-model is organised according to knowledge dimensions. A dimension denotes an aspect or a perspective from which to consider the enterprise in question. As such, the dimensions define conceptual domains, each with their own set of constructs to be used for modeling the enterprise. Presently, five dimensions are included in the POP*, of which the three first are explicitly referred to by the ‘POP*’ acronym. The ‘*’ refers to additional dimensions (an undetermined number of dimensions may be included). So far at the current stage of development, the following dimensions are included in POP* meta model: Process, Organisation, Product, Decision, and Infrastructure dimensions. The modeling constructs of product dimension is shown figure below.

Other interesting dimensions are Organisation dimension and Decision dimension. The Decision dimension modeling constructs are shown figure below.
An example of use of POP* as a neutral format to develop interoperability of enterprise models is shown in figure below. This situation is characterized by two or more enterprises using different Enterprise Modeling Languages (GRAI, IEM, ARIS Language, UML, etc.) with their corresponding tools (GraiTools, MO2GO, Metis, ARIS Process Platform, Rational, etc.). They need to exchange their complete or partial enterprise models to be able to collaborate.

There are also several real applications of the POP* experimented in German industry.

Supporting tool
A prototype supporting tool was developed by IPK (Berlin) under ATHENA project.

Remarks
The POP* is developed at origin to not only to model an enterprise as a modeling language but also as a neutral format to allow mapping between different modeling languages for interoperability purpose POP* is also supported by a methodological guide which provides instructions on how to apply POP* in enterprise.
Product dimension modeling constructs can be used in MSEE to model product at BOM level

Open issue

Reference
- ATHENA, DA1.3.1, Report on Methodology description and guidelines definition, ATHENA project deliverable, 2005.
15.5. EN/ISO 19440

<table>
<thead>
<tr>
<th>Name</th>
<th>EN/ISO 19440 – Constructs for Enterprise Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>CEN TC310/WG1 and ISO TC184/SC5/WG1</td>
</tr>
</tbody>
</table>

**Objective**

The standard EN/ISO 19440 aims at defining a set of conformant modeling language constructs which provide common semantics and enable the unification of models developed by different stakeholders in the various phases of model development.

**Background**

European Standardization in the field of Advanced Manufacturing Technology is undertaken by CEN TC310, Advanced Manufacturing Technology. In turn, its working group CEN TC 310 WG1, “Systems Architecture”, hereafter referred to as TC310 WG1, is concerned with standardization work in the field of CIM Systems Architecture. This work is to be a precursor and a contribution to the development of CEN and ISO standards in this area.

Main modeling concepts are based on CIMOSA modeling framework and modeling constructs, developed by AMICE consortium under the framework of several European projects since 1985.

**Description**

EN/ISO 19440 standard identifies and defined enterprise modeling concepts and constructs from four Enterprise Model Views: Function, Information, Resource, Organisation, to allow the modeling of the major aspects of an enterprise. Additionally, other views may be defined if needed and supported by the engineering tools, such as decision view, purpose views, and implementation views.

The complete set of modeling language constructs (without their associations) is shown in Figure below. Note that although Relationship is shown on this diagram, it is not a construct.

Figure below shows the organisation view modeling constructs and their relationships to other view’s constructs. It illustrates how the various constructs are managed by Organisational Units and Organisational Cells/Decision Centres.
Some supporting tools were developed to support CIMOSA modeling language, such as ARIS 6 Collaborative Suite, FirstSTEP 4 Toolset (Interfacing Technologies).

Main application of this approach is partly represented by ARIS process modeling approach.

15.6. Blueprint

<table>
<thead>
<tr>
<th>Name</th>
<th>Blueprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>Shostack (1)</td>
</tr>
<tr>
<td>Objective</td>
<td>Blueprinting is a method to model service processes.</td>
</tr>
</tbody>
</table>

**Background**

Service Blueprint is defined as “a picture that portrays the service system so that the different people involved in its development can understand and deal with it objectively regardless of their roles or their individual point of view” [4]. It depicts the process of service delivery, the role played by the actors of the process: customers, service employees, and visible components of service. Service Blueprint answers these questions: Who does What, To Whom, How often, under what conditions.

Blueprint is a two-dimensional picture of service process. The horizontal axis represents the chronology of actions conducted by the customer and provider of the service. The vertical axis distinguishes between different areas of action separated by different lines. In reference to the application of blueprinting in different purposes in the latest years, three stages of development separated by lines can be identified:

- **Line of interaction**: separates the customer’s actions area from the supplier’s action area. Above this line, we find activities, choices, needs and interactions performed by customers.
- **Line of visibility**: distinguish between actions visible and invisible for customers. Above this line one can find the “On-stage” contact and employees’ actions (Front Office).
- **Line of internal interaction**: distinguishes between front office and back office activities.

Above this line one can find the “Backstage” contact and employees’ actions.

**Support processes**: processes necessary for delivering the service and carried out beneath the line of internal interaction.

**Example**

Supporting tool

The traditional Blueprint of the third stage was revised by certain authors [2; 3; 4] in [7]. They said that it is not homogeneously structured and have proposed two additional action areas separated by horizontal lines.

- **The line of implementation**: separates management zone (planning, managing and controlling) and support zone (support activities)
- **The line of order penetration**: beneath the line of internal interaction,
consumer-induced from customer-independent activities. The integration of two additional lines is the result of the lack of differentiation between customer-induced and customer-independent activities in the traditional blueprint. The revised blueprint [24] shows activities concerning the production structure of service operations and refers rather to the value chain of services than to the structure of service process [5].

Service Blueprint simplifies service complexities by displaying the operation of existing system. When the current operation is explicit, managers can make rational choices about how they will operate in the future. Blueprints can assist Business unit managers in the decision making activities associated with strategy setting, allocation of resources, integration of service functions and evaluation of performance overall. It is also useful for marketing and communication people as a guide to the key service components contributing to consumers’ satisfaction and for human resource managers in the preparation of job descriptions [6].

<table>
<thead>
<tr>
<th>Open issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Shostack, G., &quot;How to Design a Service.&quot; European Journal of Marketing, 161982, 49-63</td>
</tr>
</tbody>
</table>

Reference
## 15.7. BPMN

<table>
<thead>
<tr>
<th>Name</th>
<th>BPMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>OMG</td>
</tr>
</tbody>
</table>

### Objective

The Object Management Group (OMG) has developed a standard Business Process Model and Notation (BPMN). The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.

### Background

Business Process Model and Notation (BPMN) is a graphical representation for specifying business processes in a business process model. It was previously known as Business Process Modeling Notation. Business Process Management Initiative (BPMI) developed BPMN, which has been maintained by the Object Management Group since the two organizations merged in 2005. As of March 2011, the current version of BPMN is 2.0.

### Description

BPMN models consist of simple diagrams constructed from a limited set of graphical elements. For both business users and developers, they simplify understanding business activities' flow and process. BPMN's four basic element categories are:

- Flow objects: Events, activities, gateways
- Connecting objects: Sequence flow, message flow, association
- Swim lanes: Pool, lane
- Artifacts: Data object, group, annotation

These four categories enable creation of simple business process diagrams (BPDs). BPDs also permit making new types of flow object or artifact, to make the diagram more understandable.

### Example

In modelio, BPMN models are connected to UML models when necessary.

Example of a BPMN model

### Supporting tool


### Remarks

BPMN is more formal than IDEF3 that is more adapted for business user

### Open issue

- None

### Reference

15.8. Smart network modeling method

<table>
<thead>
<tr>
<th>Name</th>
<th>Smart network modeling method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>IAO Stuttgart &amp; DITF-MR Denkendorf, Germany</td>
</tr>
</tbody>
</table>
| Objective                   | Main objective of Smart network modeling method aims at:  
|                             | - Description of all areas in an Ecosystem that can be directly influenced e.g. via services  
|                             | - Description of Smart Organisations/Smart Networks in all its dimensions |
| Background                  | Smart networks are networks of legal entities. The concept involves the design, implementation and the upkeep of such networks considering organisational, ICT and knowledge aspects. Other aspects like legal or financial aspects are not covered by the concept but to a certain degree by the modeling method |
| Description                 | Smart network modeling method proposes to build 5 types of models as shown figure below. |

![Diagram of Smart network modeling method](image)

Following a top-down approach the first model type used will be the ecosystem model. This model type describes the network and its environment including the stakeholders as well as the goods and processes. The second model type is the process model. It provides more details like sequence of activities or used resources about the identified processes in the ecosystem model. The last model type (resource model) describes the infrastructure of the network, the capabilities and dependencies of resources. Beside this top-down sequence of model two other model types exist. The first model type describes the information landscape of the network and the second one the knowledge landscape. Figure below shows modeling concepts mapped to the CIM level.
<table>
<thead>
<tr>
<th>Example</th>
<th>The following shows an example of Bivolino case modeled using the Smart Network Method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supporting tool</td>
<td>The smart network modeling method is implemented in the ‘Generic Modeling Environment’ (GME)(^{15}) of the Vanderbilt University. It is Open Source (Meta) Modeling Environment.</td>
</tr>
</tbody>
</table>

\(^{15}\) See (Ledeczi, et al., 2001)
GME is a configurable application for creating domain-specific modeling environments. The configuration of the application is performed with a meta-modeling language (MetaGME) similar to Meta Object Facility (MOF) that means it is based on UML class diagram notation and uses the Object Constraint Language (OCL) to describe constraints.

<table>
<thead>
<tr>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>- The business level (CIM) of the SmartNets Modeling comprises; namely SmartNets (ICT, Organisation, ICT) and Networking (Finances, Customer, Product/Service, Process/Activity, Actor, ICT) perspective.</td>
</tr>
<tr>
<td>- The second layer (PIM) deals with service retrieval, orchestration, and usage issues. Interfaces between service and production systems are defined and semantically enriched.</td>
</tr>
<tr>
<td>- In the TSM layer, platform specific aspects of services can be modeled and defined. Depending to the to-be runtime environment of a service, the service logic has to be detailed differently.</td>
</tr>
</tbody>
</table>

| Open issue |
| Reference |
## 15.9. SmartService

<table>
<thead>
<tr>
<th>Name</th>
<th>SmartService (Ontology-driven Service Development (OSD) method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>DITF-MR Denkendorf, Germany</td>
</tr>
<tr>
<td>Objective</td>
<td>The objective is to specify ontology-based services with the help of semantic models (that can directly be interpreted by reasoning engines).</td>
</tr>
<tr>
<td>Background</td>
<td>OSD derives from the Ontology Driven Architecture idea where ontology engineering processes and software/service engineering principles are merged: „Hence, given the semantically rich, unambiguous qualities of information embodiment on the Semantic Web, the amenable syntax of Semantic Web languages, and the universality of the Semantic Web's XML ancestry, there appears a compelling argument to combine the semi-formal, model driven techniques of Software Engineering with approaches common to Information Engineering on the Semantic Web” (W3C, 2006).</td>
</tr>
<tr>
<td>Description</td>
<td>SmartServices are specified and (graphically) modeled by the help of formal semantics. The resulting models can easily be interpreted by respective inferencing mechanisms (as runtime environment). Accordingly, no programming, compilation, or even software code is necessary to “implement” (deploy) SmartServices. Formal semantics can be expressed using ontologies as a “formal specification of a shared conceptualization”. Formal ontologies comprise concepts (core meanings of nouns), attributes (attributes of concepts), relations (relations between concepts and instances), as well as instances (instantiations of concepts) and rules (to be applied to all ontological artifacts). In combination with MDA’s third layer of abstraction, Ontology-driven Architecture (ODA) provides a middle-out methodology to capture all required information to model and specify SmartServices, starting with formal performance questions, then contextualization of the service (the problem expressed in performance question), and finally modeling its internal functionalities (generic solution, provided in knowledge base of the ecosystem). Figure below shows a comparison between MDA and ODA (Hirsch, 2012).</td>
</tr>
<tr>
<td>Example</td>
<td>In MSEE ontologies are modeled using an ODA-based Ontology-driven Service Development (OSD) method by first order logic-based expressions as well as graphical representations that can be transformed 1:1. An example for this transformation is</td>
</tr>
</tbody>
</table>
as follows:

**English:**

"Who should work with whom?"

**Graphical:**

![Diagram](image)

**Formal:**

\[
\text{FORALL anActor,anActor1} \\
\text{anActor[actorMightSetUpProjectWith-> anActor1] } \\
\text{- anActor:#Actor AND} \\
\text{anActor1:#Actor.}
\]

Performance Question in English and first order logic (graphical and textual).

---

**Supporting tool**

To support the development of SmartServices as well as the transformation of graphical to textual semantic models, dedicated software tools can be used that are both, open source (e.g. protégé) and commercially available (e.g. OntoBroker of Ontoprise GmbH, Karlsruhe). The latter explicitly supports standard ontology modeling techniques including rules. Graphical models can thus be transformed and encapsulated in standard WSDL-compatible web-services format and accessed online.

---

**Remarks**

ODA is not considered to be a substitute for MDA but a complementary approach that is applied in the third abstraction layer of MDA to define SmartServices. To operationalize the ODA idea for SmartServices an adapted software development methodology according to IEEE 1074-2006 can be applied – from analysis, to specification, conceptualization & formalization, deployment, and usage & maintenance. These development steps are combined with additional supporting activities like evaluation, documentation, knowledge acquisition, and integration (of existing instances for example).

---

**Open issue**

A pure technological issue remains; even though the same meta-model applies, OntoBroker and OntoStudio (Ontoprise 2008) distinguish between ontology-model and rule-model that have to be merged while deploying a SmartService. This gap has to be bridged in order to ease the service development process.

---

**Reference**


## 15.10. USDL

<table>
<thead>
<tr>
<th>Name</th>
<th>USDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>Initially USDL has been developed by SAP Research and it has been developed within the USDL community (<a href="http://www.internet-of-services.com/index.php?id=264">http://www.internet-of-services.com/index.php?id=264</a>). Standardisation of USDL has been investigated within W3C’s Unified Service Description Language Incubator Group (<a href="http://www.w3.org/2005/Incubator/usdl/charter">Link: http://www.w3.org/2005/Incubator/usdl/charter</a>)</td>
</tr>
<tr>
<td>Objective</td>
<td>Unified Service Description Language (USDL) is designed as a language for describing general and generic parts of technical and business services to allow services to become tradable and consumable. The USDL specification aims at complementing the technical language stack by adding required business and operational information. Technical services are considered electronic services based on WSDL, REST or other technical specifications. Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer. Thus, the business services are more general and comprise manual and technical services. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers. The language is usable for any purpose and implementation scenario of future business services on a general level. However, it will be extendable for industry-specific aspects. ([Direct citation from Source: <a href="http://www.w3.org/2005/Incubator/usdl/charter%5D">http://www.w3.org/2005/Incubator/usdl/charter</a></td>
</tr>
</tbody>
</table>
| Background | A gap analysis of current state of the art for service modeling languages led to the following result: A service description language is missing, which emphasizes the business-related aspects when defining a service. Service description languages such as WSDL concentrate on the proposition of technical interfaces to get services exchanged. Stakeholders from small, medium and large sized businesses as well as representatives from various industries expressed the need of an open and accessible service description language that is able to describe their business needs based on the following aspects (modules):  
  - Core (general information about the service)  
  - Interaction (how the service can be consumed)  
  - Participant (roles of businesses and / individuals that become involved in the provision and consumption of a service)  
  - Functional (what business functions are provided by the service)  
  - Pricing (how much does the service cost)  
  - Legal (legal terms and conditions under which the service may be consumed)  
  - Service Level Agreements (levels of service provided, e.g. availability, response time, etc.) ([Direct citation from Source: [http://www.w3.org/2005/Incubator/usdl/charter](http://www.w3.org/2005/Incubator/usdl/charter]) |
| Example |  
| Supporting tool | - A USDL editor has been developed as Open Source software ([http://www.internet-of-services.com/index.php?id=290](http://www.internet-of-services.com/index.php?id=290)).  
- A USDL repository has also been developed for storage/retrieval of USDL files as XML-files. |
| Remarks |  |
| Open issue |  |
### 15.11. WSDL

<table>
<thead>
<tr>
<th>Name</th>
<th>WSDL (Web Services Description Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>World Wide Web Consortium (W3C)</td>
</tr>
</tbody>
</table>

**Objective**

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

**Background**

As communications protocols and message formats are standardized in the web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

**Description**

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types** – a container for data type definitions using some type system (such as XSD).
- **Message** – an abstract, typed definition of the data being communicated.
- **Operation** – an abstract description of an action supported by the service.
- **Port Type** – an abstract set of operations supported by one or more endpoints.
- **Binding** – a concrete protocol and data format specification for a particular port type.
- **Port** – a single endpoint defined as a combination of a binding and a network address.
- **Service** – a collection of related endpoints.

A WSDL document is simply a set of definitions. There is a definitions element at the root, and definitions inside. Services are defined using six major elements:

- **types**, which provides data type definitions used to describe the messages exchanged.
- **message**, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
- **portType**, which is a set of abstract operations. Each operation refers to an input message and output messages.
- **binding**, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
- **port**, which specifies an address for a binding, thus defining a single communication endpoint.
- **service**, which is used to aggregate a set of related ports.

The following figure shows WSDL metamodel represented in UML notation.

---

**Example**

**Supporting tool**

At TSM, the Web Services Description Language (WSDL) is an XML-based language that is used for describing the functionality offered by a Web service. A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects and what data structures it returns. It thus serves a roughly similar purpose as a Method signature in a programming language.

**Remarks**

Under MDA, WSDL can be at PIM and/or PSM level. It depends on the perspective you choose. If for instance let’s say Java as your platform of choice is at PSM level then WSDL would be at PIM level. The same applies to Java that can be considered at PIM level with respect to different operating systems that would be at the PSM level.

**Open issue**

- V. de Castro, E. Marcos and B. Vela, Representing WSDL with Extended UML.

**Reference**
# 15.12. REST

<table>
<thead>
<tr>
<th>Name</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>REST is not a protocol or format it is an architecture style, in fact the original architectural style of the Web.</td>
</tr>
<tr>
<td>Objective</td>
<td>REST is a Web services architecture, like SOAP and XML-RPC. This is an acronym for Representational State Transfer. Developed in 2000 by Roy Fielding, one of the creators of the HTTP Server Apache HTTPd and other fundamental work, REST is behind an attempt to describe the principles of Web architecture.</td>
</tr>
<tr>
<td>Background</td>
<td>The REST architecture has been developed in parallel with the protocol HTTP/1.1.</td>
</tr>
<tr>
<td>Description</td>
<td>REST can take into account the behaviour and the relation with the service client. REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource. At any particular time, a client can either be in transition between application states or &quot;at rest&quot;. A client in a rest state is able to interact with its user, but creates no load and consumes no per-client storage on the set of servers or on the network.</td>
</tr>
</tbody>
</table>
| Supporting tool | - jrest  
- restlet  
- cxf  
- jersey  
- axis2  
- Spring3  
- Resteasy |
| Remarks | - Fielding’s dissertation "Representational State Transfer (REST)".  

---

**Diagram:**
- Editors: [http://wiki.restlet.org](http://wiki.restlet.org)
### 15.13. BPEL

<table>
<thead>
<tr>
<th>Name</th>
<th>BPEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>OASIS standard executable language</td>
</tr>
<tr>
<td>Objective</td>
<td>The BPEL, Business Process Execution Language is an XML representation used in the framework of establishment of a Service-Oriented Architecture (SOA) in IT business. Specifically, in the SOA, the enterprise applications are encouraged to conform to a common reference to promote dynamic dialogue between existing applications and consolidate them. BPEL just holds the dialogue between the different applications of SOA.</td>
</tr>
<tr>
<td>Background</td>
<td>BPEL comes in the form of an XML file; that is readable by engine business process management to enforce the rules of the exchanged files. Concretely, without BPEL, Web services could not talk literally to each other. It then organizes the conduct of business processes (workflow) including causality and time management. The BPEL file thus acts on elements such as data transformation, sending messages or calling a function. BPEL uses WSDL to describe the actions of a process. This language shows all the elements necessary to interact with a network service (available functions, protocols, service address). It also uses SOAP, the standard protocol for Web services, to define how to structure the messages will exchange applications.</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td><img src="image.png" alt="BPEL Diagram" /></td>
</tr>
<tr>
<td>Source</td>
<td><a href="http://eclipse.org/bpel/">http://eclipse.org/bpel/</a></td>
</tr>
<tr>
<td>Supporting tool</td>
<td>BPEL designer: <a href="http://eclipse.org/bpel/">http://eclipse.org/bpel/</a></td>
</tr>
<tr>
<td>Remarks</td>
<td>BPEL uses such as UDDI standards to connect to corporate directories. These directories referencing the enterprise corporate network and all existing Web services.</td>
</tr>
<tr>
<td>Open issue</td>
<td>When the orchestration becomes too complex, for example when to choose which version of a Web service in a directory based on the execution context of the process.</td>
</tr>
<tr>
<td></td>
<td>- Open Source Easy BPEL / Petals BPEL Engine <a href="http://easybpel.petalslink.org/user-guides.html">http://easybpel.petalslink.org/user-guides.html</a></td>
</tr>
</tbody>
</table>