Contribution to a methodology for service systems modeling and engineering through a model driven approach : architecture, transormation, and model simulation

Hassan Bazoun

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGENIEUR

SPÉCIALITÉ PRODUCTIQUE

Par Hassan BAZOUN

Contribution to a methodology for service systems modeling and engineering through a model driven approach

Architecture, transformation, and model simulation

Sous la direction de : Yves Ducq
(Co-directeur : Gregory Zacharewicz)

Soutenue le 20 octobre 2015

Membres du jury :

M. CHEN, David	Professeur Université de Bordeaux	Président
M. CHAPURLAT, Vincent	Professeur Ecole des Mines d'Alès	Rapporteur
M. TRAORE, Mamadou Kaba	Maître de Conférences Université Blaise Pascal	Rapporteur
M. DUCQ, Yves	Professeur Université de Bordeaux	Directeur de thèse
M. ZACHAREWICZ, Gregory	Maître de Conférences Université de Bordeaux	Co-directeur de thèse
M. BOYE, Hadrien	Ingénieur, HARDIS Group	Chef de projet
M. SROUR, Zein	Docteur, Ancien directeur de HARDIS Ouest	Invité

,

Titre :

Contribution à une méthodologie pour la modélisation des systèmes de services et d'ingénierie grâce à une approche dirigée par les modèles: l'architecture, la transformation et la simulation du modèle.

Résumé :

Cette thèse se situe dans le contexte de l'importante mutation stratégique qu'opère l'Industrie européenne face à l'émergence de nouveaux Marchés. Une caractéristique majeure de ces nouveaux Marchés est la grande variabilité des besoins clients. Cette mutation remplace le produit manufacturé, au cœur des stratégies Métier, par ses services d'accompagnement, en réponse aux nouvelles exigences des clients. Ainsi, les processus Métier, initialement pensés, construits et pilotés autour du produit, doivent aujourd'hui être revus et complétés de manière à intégrer les services. C'est cette question que veut traiter la thèse, à travers une proposition d'architecture d'ingénierie des services dirigée par les modèles, supportée par un environnement logiciel appelé SLMToolBox qui permet la semi automatisation d'une partie de la gestion du cycle de vie d'un service (modélisation, simulation et évaluation de performances). Ce travail de recherche était dans le cadre de projet MSEE, un projet européen de recherche et développement en collaboration avec 18 partenaires de 9 pays européen. Le but de ce projet est de faire évoluer le concept de SSME (Service Science Management and Engineering) vers des systèmes de production et des usines du futur, i.e. d'un point de vue méthodologique, pour adapter, modifier et étendre les concepts de SSME pour les rendre applicables à des entreprises traditionnellement orientées vers une production orientée produit et d'un point de vue implantation, d'instancier les architectures et les plateformes orientées vers les services liés au futur internet pour des systèmes globaux de production de services.
La thèse à apporter plusieurs résultats (MDSEA, Etended Actigram Star EA*, Transformation de modele, simulation, et SLMToolBox) pour répondre aux besoins de servitization.
Le MDSEA apporte un cadre méthodologique générique inspiré de l'Ingénierie Dirigée par les Modèles et dont le bénéfice principal est de permettre d'exprimer le traitement de toute question relative au cycle de vie d'un service, au travers de modèles spécifiés à divers niveaux d'abstraction, et reliés entre eux par des mécanismes de transformation de modèle.
Chacun de ces niveaux de modélisation nécessite des langages de modélisation spécifiques. Cette architecture suggère 3 niveaux d'abstraction : (1) un niveau appelé BSM (pour Business Service Model) où sont spécifiés les modèles conceptuels de processus Métier orientés Service à l'aide d'un langage conçu à cet effet, nommé EA* (pour Extended Actigram Star) et inspiré du langage GRAI Extended Actigram ; (2) un niveau appelé TIM (pour Technology Independent Model) où sont spécifiés les modèles détaillés de ces mêmes processus à l'aide du langage BPMN (Business Process Modeling Notation), modèles obtenus par transformation de modèle EA* en BPMN ; et (3) un niveau appelé TSM (pour Technology Specific Model) où sont spécifiés les modèles avec les choix technique spécifique au développement et génération de code. MDSEA ne se limite ainsi pas aux seuls aspects liés aux IT mais aussi aux aspects liés ressources humaines et matérielles devant être prises en compte, créées ou encore achetées pour mettre en œuvre le service attendu et le gérer au long de son cycle de vie.

Extended Actigram Star (EA*) est un langage de modélisation de processus business, développé dans le cadre de cette thèse et inspiré du langage GRAI Extended Actigram. La syntaxe abstraite et concrète de ce langage est décrite de manière détaillée.

La transformation de modèle est basée sur le « mapping » entre méta modèles. Le mapping a appliqué est défini après avoir étudié le langage source et langage cible. Puis on a implémenté les règles de transformation en utilisent « ATLAS Transformation Language » ATL. La transformation de modèle nous permet dans le cadre de MDSEA de passer d'un niveau d'abstraction vers un autre. Deux transformations des modelés sont développées pendant la thèse : EA* vers BPMN et BPMN vers DEVS.

La simulation est un outil d'assistance à l'ingénierie, comme au management ou encore au pilotage de systèmes complexes, DEVS est aussi reconnu comme un concept formalisé et largement usité pour la modélisation et la simulation du comportement de système basé sur une hypothèse de comportement à événements discrets. J'ai travaillé sur les règles de transformation de modèle BPMN 2.0 vers DEVS afin d'en permettre la simulation. Cette simulation a pour objet d'évaluer les performances en termes de coût de de temps mais évoque aussi des performances en termes de qualité et de flexibilité. Ces transformations sont à la base d'une sémantique opérationnelle de BPMN i.e. des règles, éventuellement assorties de probabilités d'évolution, décrivant comment un concept de modélisation interagit avec d'autres concepts et évolue en conséquence d'un état au suivant si l'on adopte une vision dynamique basée sur un modèle à états / transitions et événements comme DEVS. Le modèle finale DEVS est simulé au travers des profils de simulation.

L'outil SLMToolBox intègre les langages adoptés, implémente les règles de transformation de modèle, et offre un environnement de simulation, d'évaluation des performances et d'animation des résultats. Il vient donc en support aux analystes Métier qui, en collaboration avec les experts des différents domaines Métier, peuvent ainsi décrire et évaluer leurs systèmes de services courants (modèles AS-IS), mais aussi concevoir et évaluer de nouveaux services (modèles TO-BE). SLMToolBox est une application Eclipse RCP (Rich Client Platform). C'était développé en java et en utilisent des « Framework » différents : EMF/Ecore (Génération de code et représentation des modèles), EEF (pour gérer les « properties » des objets graphiques), Graphiti (développement des editors graphiques), ATL (Transformation des modèles). La SLMToolBox est un des résultats apprécié dans le projet MSEE. Plusieurs réunions ont eu lieu à Bruxelles pour la création d'une communauté scientifique autour de la SLMToolBox. Le but de cette Communauté est de reprendre le développement de la SLMToolBox pour l'adapter aux besoins différents des clients dans plusieurs domaines. Grace à cette communauté, le SLMToolBox était utilisé dans plusieurs projets (comme NOSCIFel dans le domaine de transport…).

Mots clés :

Servitization, System de Service, MDSEA, Extended Actigram Star, Transformation de modèle, SLMToolBox, Simulation DEVS.

Title:

Contribution to a methodology for service systems modeling and engineering through a model driven approach: Architecture, transformation, and model simulation.

Abstract:

In today's world of business, manufacturers are facing many challenges. Business strategies

are changing and manufacturers are entering new markets and striving to meet new and changing customer needs. Manufacturers are outsourcing more components and services to suppliers around the world, restructuring their internal operating and information systems, and re-engineering production processes to eliminate waste and lower costs. They are changing the nature of their organizations by partnering with other companies in complex supply chains and business networks that now extend globally. Manufacturing is being redefined by changes in market place and how companies react to them. As a result, many manufacturers wanted to make the shift to services as a solution, but they find themselves trapped in the world of products. 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 surrounding products became more and more important in manufacturing. The process of creating value by adding services to a tangible product has first been called "servitization". Based on the problematic of Servitization and service system engineering and in order to reduce effort and time in service system engineering, this thesis (as being part of the MSEE project) contributed in the development of solutions. The contribution of the thesis's result can be classified into related and connected pillars. The first pillar is the participation in the development of the Model Driven Service Engineering Architecture (MDSEA) which permits Virtual Manufacturing Enterprises (VME) to model their service systems (AS-IS and TO-BE models) starting from modeling the system from business experts angle and then adding more details to reach the developers and technical experts angle. The second pillar is the development of a modeling and simulation tool, the SLMToolBox. This tool is a partial implementation of MDSEA and its name Service Lifecycle Management ToolBox implies a role in the service's lifecycle. The third pillar is the development of a DEVS graphical editor and simulator integrated in the SLMToolBox.

Keywords:

Servitization, Service System, MDSEA, Extended Actigram Star, Model Transformation, SLMToolBox, DEVS Simulation.

Acknowledgment

After three years working on this thesis, I would like to thank the IMS laboratory for accepting me as part of their team. A Big thanks to Dr. Yves Ducq and Dr. Gregory Zacharewicz for the support and help they offered. Without them I would not be able to finish or even start this thesis. They were always available for my questions which were many at the beginning (when I was lost on how to start) and the end during the writing of the final manuscript. Another special thanks to Hadrien Boye my project manager at Hardis Group during my three years. I learned from Hadrien a lot about organizing my work, how to build my answers and look for solutions, and most important how to always search for the bright side of our work. Without the help of Hadrien I would not be able to deliver the work with the same quality. Besides, I would like to thank my colleges at Hardis with whom we shared a lot of good moments that gave me a positive force to finish this work.

Thanks to my uncle Zein Srour and it's because of his support and help I finished my masters and PHD studies in France. I learned a lot from you on how to always keep walking under whatever circumstances, to be positive and that hard work pays at the end. You were the source of motivation and light when the journey started to get darker. Thank you for everything you have done for me. Also thanks to all members of my uncle's family at Nantes for the good times we spent together.

Being a PHD student is not always about being late at work trying to keep it up with deadlines for scientific articles or finding solutions after weeks of being moving around in circles. It's also about having good friends with whom you spend funny, happy and relaxed moments. Thanks to Ali, Gilberto, Elise, Yaaroub, Varvara, Chouppi, Homam, Charlene, Joe, Bianca, Mouna… and from Lebanon Jana, Tigo, Nassim, Ali B., Ali W., Yazback…

A special Thanks to Clara, the one who gave me everything, she was my best friend, the love in my life and the family I needed. She stood beside me in the darkest moments of the journey and helped me to overcome it with success.

Last but not least, a big thanks to my family in Lebanon, my parents (Fatmeh and Hussein), Dima, Mariam, Amer, Mohammed, Mano, uncles, aunts and cousins for all the love and support you gave me and are still giving.

This thesis is dedicated to my mother for everything she has done and for here unconditional love, to the memory of my grandmother who left us one month before defending my thesis, and to the memory of my father who can rest now after I finished what he started.

GENERAL INTRODUCTION AND PROBLEM TO SOLVE: FROM SERVICE TO SERVICE MODELING IN VIRTUAL MANUFACTURING ENTERPRISE CONTEXT 1

1. CONTEXT 1

2. PRINCIPLES OF SERVICE AND SERVICE SYSTEM MODELING 2

2.1 FROM SERVICE TO SERVITIZATION 2
2.1.1 Characterization of a service 3
2.1.2 Product service and PSS 3
2.2 MSEE SERVITIZATION CONCEPTS 4
2.2.1 Extended Product (EP) 4
2.2.2 Product+Service and Product2Service 5
2.2.3 Service Life cycle Management 5
2.3 FROM ENTERPRISE TO MANUFACTURING SERVICE ECOSYSTEM 6
2.3.1 From a single Manufacturing Enterprise to a Virtual Manufacturing Enterprise 6
2.3.2 From Virtual Manufacturing Enterprise to Manufacturing Service Ecosystem 7
2.4 SERVICE SYSTEM AND SERVICE SYSTEM LIFE CYCLE MANAGEMENT 8
2.4.1 Service System 8
2.4.2 Service System Life cycle Management (SLM) 9
2.4.3 Servitization and Service System evolution 11
2.5 MODELING OF SERVICE SYSTEM 12
2.5.1 Why to model Service System? 12
2.5.2 Modeling Service System using System Theory 13
2.5.3 Definition of the languages to describe and represent the Models 15
2.6 ARCHITECTURE FOR SERVICE SYSTEM ENGINEERING 17
2.6.1 Service system engineering 17

3. MANUFACTURING SERVICE ECOSYSTEM (MSEE) PROJECT 17

3.1 MSEE RESULTS 18
3.1.1 MSEE Generic Assets 18
3.1.2 MSE-Specific Assets 18
3.1.3 VME-Specific Assets 19

4. CONTRIBUTION OF THE THESIS 19

5. ORGANIZATION 20

STATE OF THE ART 22

1. ENTERPRISE MODELLING 23

1.1 CIMOSA 23
1.1.1 Approach 23
1.1.2 Overview 24
1.2 GIM 26
1.2.1 GIM phases 26
1.3 ARIS 29
1.3.1 Concept of ARIS architecture 29
1.4 CONCLUSION ON ENTERPRISE MODELING 29

2. ENTERPRISE INTEROPERABILITY 30

2.1 DEFINITIONS 30
2.2 DIMENSIONS 31
2.3 APPROACHES AND FRAMEWORKS 32
2.3.1 IDEAS interoperability framework 32
2.3.2 LISI approach 34
2.3.3 ATHENA interoperability framework 35
2.4 CONCLUSION ON ENTERPRISE INTEROPERABILITY 37

3. MODEL DRIVEN DEVELOPMENT 38

3.1 MDA 38
3.1.1 Overview 38
3.1.2 MDA for Reuse and Interoperability 39
3.2 MDI 40
3.3 Conclusion on Model Driven Development 41

4. MODELLING LANGUAGES 42

4.1 GRAI Extended Actigram 42
4.2 BPMN 42
4.3 DEVS Formalism 44
4.3.1 Atomic DEVS 44
4.3.2 Coupled DEVS 45
4.4 Conclusion on Modeling Languages 45

5. SIMULATION TOOLS 46

5.1 Business Process Simulation Tools 46
5.1.1 ARIS Simulation 46
5.1.2 Protos 46
5.1.3 Arena 47
5.1.4 Jbpm 47
5.1.5 Bonita Open Solution 48
5.1.6 Evaluation 48
5.2 DEVS Simulation Tools 49
5.3 Conclusion on Simulation Tools 49

MODEL DRIVEN SERVICE ENGINEERING ARCHITECTURE (MDSEA), EXTENDED ACTIGRAM STAR (EA*), AND MODEL TRANSFORMATION 51

1. SERVICE SYSTEMS' MODELING AND MODEL DRIVEN APPROACH 52

2. MDSEA 52

2.1 Business Service Model (BSM) 55
2.2 Technology Independent Model (TIM) 55
2.3 Technology Specific Model (TSM) 55
2.4 Proposed Modelling Languages 56

3. EXTENDED ACTIGRAM STAR (EA*) 57

3.1 Scope 57
3.2 Overview 58
3.3 Abstract Syntax 58
3.3.1 Structure 59
3.4 Graphical Representations and Notations 71
3.5 Connectivity constraints 74

4. MODEL TRANSFORMATION 75

4.1 Problem 75
4.2 MetaModel Approach 76
4.3 Mapping of Concepts 78
4.3.1 Results in the frame of MDSEA 78
4.3.2 Results outside the frame of MDSEA (Generalization) 83
4.4 Example 86

5. CONCLUSION 90

SIMULATION AND MODEL TRANSFORMATION FROM BPMN TO DEVS 91

1. INTRODUCTION 92

2. PROBLEM 93

3. DEVS 94

3.1 Basic DEVS characteristics 94
3.2 Simulation of DEVS Model 94

4. TRANSFORMATION BPMN TO DEVS 95

4.1 DEVS Metamodel 96
4.2 Transformation rules 96
4.2.1 BPMN Task to DEVS Atomic Model 97
4.2.2 BPMN Event to DEVS Atomic Model 98
4.2.3 BPMN Gateway to DEVS Atomic Model 103
4.2.4 BPMN Lane, Pool, and SubProcess to DEVS Coupled Model 105
4.2.5 BPMN Flow to DEVS Coupling 105

5. DEVS SIMULATION 105

5.1 Execution 106
5.2 Simulation's profile and results 108
5.3 Animation 110

6. EXAMPLE 112

7. CONCLUSION 114

SLMTOOLBOX 115

1. SYSTEM OVERVIEW 116

1.1 Context and purpose 116
1.2 System vision and top level requirements 117
1.3 Logical architecture 118
1.4 Actors and roles 119
1.4.1 Business Actors 119
1.4.2 Domain Specific Actors 120
1.5 End-to-end scenarios 120
1.5.1 Scenario 1: Design a new service within a single enterprise 120
1.5.2 Scenario 2: Design & deploy a new service within a VME 121

2. TECHNICAL OVERVIEW 122

2.1 Technical modules 122
2.2 Application modules 124

3. IMPLEMENTATION OF MDSEA IN THE TOOLBOX 126

3.1 Modelling architecture overview 126
3.2 Service modelling features 127
3.2.1 Summary of modelling editors 127
3.2.2 GraiGrid Editor 128
3.2.3 ExtendedActigramStar Editor (BSM Level) 129
3.2.4 UML Editor 129
3.2.5 BPMN Editor 130
3.3 Model transformation features 130

4. IMPLEMENTATION OF SIMULATION IN THE TOOLBOX 131

4.1 DEVS Editor 131
4.2 Simulation profile 132
4.3 Simulate DEVS model 132
4.4 Animate DEVS diagram 133
4.5 Simulation report 133

5. INDESIT USE CASE 134

5.1 The Use case experience: from Products to Services (AS-IS Situation) 134

5.2 THE PRODUCT+SERVICE IDEA: THE CAREFREE WASHING SERVICE (TO-BE SITUATION) 138
5.3 SERVICE FUNCTIONALITIES 139
5.4 NEW ECOSYSTEM FOR TO-BE SITUATION (VE) 140
5.5 SCENARIOS AND OBTAINED MODELS 141
5.5.1 Design a single service for Indesit 142
5.5.2 Design a composite service within the Indesit VME 146
5.6 CONCLUSION ON THE INDESIT USE CASE 152
6. CONCLUSION 153
GENERAL CONCLUSION AND PERSPECTIVES 154
1. GENERAL CONCLUSION 155
2. PERSPECTIVES 158
REFERENCES 159
ANNEX-1-METAMODELS 167
ANNEX-2-SIMULATION REPORT 169
ANNEX-3-ATL AND XSLT CODE 175
ANNEX-4 USE CASE DIAGRAMS 179

Figure 1 The Extended Product Concept, adopted from (Thoben et al. 2001) 4
Figure 2 Servitization process 5
Figure 3 Service Life Cycle 6
Figure 4 Virtual Manufacturing Enterprise 7
Figure 5 Business Ecosystem Concept 7
Figure 6 Manufacturing enterprise vs. service in manufacturing virtual enterprise 8
Figure 7 Service delivery system 9
Figure 8 Service System lifecycle phases vs. Service System life - adapted from Bernus (1995) 10
Figure 9 Virtual Manufacturing Enterprise in different phases of SLM 11
Figure 10 The structure of a system 14
Figure 11 MSEE as a system of systems 15
Figure 12 Enterprise Modelling mapped to the OMG 4 level architecture 16
Figure 13 CIMOSA Modelling Approach 25
Figure 14 GIM approach 27
Figure 15 Order of models realization 28
Figure 16 Enterprise Interoperability Framework 32
Figure 17 IDEAS Interoperability Framework 34
Figure 18 LISI reference model 35
Figure 19 Athena Interoperability Reference Architecture 36
Figure 20 Structure of the AIF 37
Figure 21 OMG's Model Driven Architecture 39
Figure 22 Reference model for MDI 41
Figure 23 GRAI Extended Actigram for 'Painting Check Process' 42
Figure 24 The MDSEA architecture applied in a service network of two enterprises 54
Figure 25 MDSEA vs MDA 55
Figure 26 Abstract Syntax of Extended Actigram Star 59
Figure 27 BaseElement 60
Figure 28 Process UML object diagram 61
Figure 29 Flow 62
Figure 30 ControlFlow 63
Figure 31 SupportFlow Example 64
Figure 32 ExtendedActivity 66
Figure 33 Resource 67
Figure 34 LogicalOperator 68
Figure 35 Connector 70
Figure 36 check material quality example 70
Figure 37 Transformation architecture of EA* to BPMN 77
Figure 38 Different steps for transformation 77
Figure 39 BSM Modelling strategy 79
Figure 40 EA* to BPMN collaboration 79
Figure 41 EA* e-marketplace purchase process 88
Figure 42 BPMN e-marketplace purchase process 89
Figure 43 Relation Simulator-Model (b) 106
Figure 44 DEVS Message 107
Figure 45 Simulation Algorithm 108
Figure 46 Calculating probabilities 109
Figure 47 Calculating time and cost 110
Figure 48 Animation simplified algorithm 111

Figure 49 Animation feature 112
Figure 50 DEVS e-marketplace purchase process 113
Figure 51 initialize simulation profile 113
Figure 52 SLMToolBox - Context within the service lifecycle 116
Figure 53 Purpose: phases of the service lifecycle to support 117
Figure 54 SLMToolBox Logical Architecture 118
Figure 55 System Actors 119
Figure 56 Design a new service within a single enterprise 120
Figure 57 Design & deploy a new service within a VME 121
Figure 58 Modelling Environment - Technical Architecture Overview 124
Figure 59 Application modules 125
Figure 60 Modelling architecture's overview 127
Figure 61 Create new wizard 131
Figure 62 DEVS editor 132
Figure 63 Simulation profile 132
Figure 64 Simulation 132
Figure 65 Animation 133
Figure 66 Indesit Ecosystem 137
Figure 67 Use case AS-IS Servitization level 137
Figure 68 Use Case TO-BE Servitization level 139
Figure 69 Overall service concept (EA*) 142
Figure 70 General view (EA*) 143
Figure 71 High level system architecture (UML) 144
Figure 72 Care free washing machine website (UML) 145
Figure 73 WM usage data (UML) 146
Figure 74 Smart Detergent Provisioning (EA*) 147
Figure 75 Smart Detergent Provisioning (GraiGrid) 148
Figure 76 Performance indicators (1) 149
Figure 77 Performance indicators (2) 149
Figure 78 Performance indicators (3) 150
Figure 79 RAW Smart Detergent Provisioning (BPMN) 151
Figure 80 Rearranged Smart Detergent Provisioning (BPMN) 151
Figure 81 Enriched Smart Detergent Provisioning (BPMN) 152
Figure 82 context, problem, and contributions 157
Figure 83 BSM Core Metamodel 167
Figure 84 TIM Core Metamodel 168
Figure 85 ATL Lazy Rule: EA* Process to BPMN Process 175
Figure 86 ATL helpers 176
Figure 87 XSLT example 1 177
Figure 88 XSLT example 2 178
Figure 89 Global view 179
Figure 90 Service Ideation process 179
Figure 91 Product-Service System Design 180
Figure 92 Product and Service Design 180
Figure 93 Service System Design 181
Figure 94 Service System Design 181

Table 1 Evaluation of Business Process Simulation Tools 49
Table 2 Model attributes 60
Table 3 BaseElement attributes 60
Table 4 Process attributes 61
Table 5 FlowElement attributes 62
Table 6 Flow attributes 62
Table 7 OutputInputFlow attributes 63
Table 8 ControlFlow attributes 63
Table 9 SupportFlow attributes 64
Table 10 FlowNode attributes 64
Table 11 ExtendedActivity attributes 65
Table 12 AtomicExtendedActivity attributes 66
Table 13 StructuralExtendedActivity attributes 66
Table 14 Resource attributes 66
Table 15 Human attributes 67
Table 16 Material attributes 67
Table 17 IT attributes 67
Table 18 LogicalOperator attributes 68
Table 19 Diverging attributes 68
Table 20 DivergingAnd attributes 69
Table 21 ConvergingAnd attributes 69
Table 22 Converging attributes 69
Table 23 DivergingOr attributes 69
Table 24 ConvergingOr attributes 69
Table 25 InternalConnector attributes 71
Table 26 ExternalConnector attributes 71
Table 27 ProcessConnector attributes 71
Table 28 Organization attributes 71
Table 29 Graphical Representations 72
Table 30 Flow Constraints 74
Table 31 EA* to BPMN - Mapping (Collaboration Diagram) 80
Table 32 EA* to BPMN - Mapping (Collaboration Diagram) 82
Table 33 EA* to BPMN - Mapping (Collaboration Diagram) 83
Table 34 BPMN Task to DEVS Atomic Model 97
Table 35 BPMN Receive Task to DEVS Atomic Model 98
Table 36 BPMN Send Task to DEVS Atomic Model 98
Table 37 BPMN Start Event to DEVS Atomic Model 99
Table 38 BPMN Message Start Event to DEVS Atomic Model 99
Table 39 BPMN Timer Start Event to DEVS Atomic Model 100
Table 40 BPMN None Intermediate Event to DEVS Atomic Model 101
Table 41 BPMN Message Intermediate throw Event to DEVS Atomic Model 101
Table 42 BPMN Message Intermediate catch Event to DEVS Atomic Model 102
Table 43 BPMN End Event to DEVS Atomic Model 102
Table 44 BPMN Message End Event to DEVS Atomic Model 103
Table 45 BPMN Multiple End Event to DEVS Atomic Model 103
Table 46 BPMN Exclusive Gateway to DEVS Atomic Model 104
Table 47 BPMN Parallel Gateway to DEVS Atomic Model 105
Table 48 BPMN SubProcess to DEVS Coupled Model 105
Table 49 BPMN Flow to DEVS 105

Table 50 SLMToolBox - Modelling editor’s overview 128

Résumé substantiel en français

Cette thèse se situe dans le contexte de l'importante mutation stratégique qu'opère l'Industrie européenne face à l'émergence de nouveaux Marchés. Une caractéristique majeure de ces nouveaux Marchés est la grande variabilité des besoins clients. Cette mutation remplace le produit manufacturé, au cœur des stratégies Métier, par ses services d'accompagnement, en réponse aux nouvelles exigences des clients. Ainsi, les processus Métier, initialement pensés, construits et pilotés autour du produit, doivent aujourd'hui être revus et complétés de manière à intégrer les services. C'est cette question que veut traiter la thèse, à travers une proposition d'architecture d'ingénierie des services dirigée par les modèles, supportée par un environnement logiciel appelé SLMToolBox qui permet la semi automatisation d'une partie de la gestion du cycle de vie d'un service (modélisation, simulation et évaluation de performances). Ce travail de recherche était dans le cadre de projet MSEE, un projet européen de recherche et développement en collaboration avec 18 partenaires de 9 pays européen. Le but de ce projet est de faire évoluer le concept de SSME (Service Science Management and Engineering) vers des systèmes de production et des usines du futur, i.e. d'un point de vue méthodologique, pour adapter, modifier et étendre les concepts de SSME pour les rendre applicables à des entreprises traditionnellement orientées vers une production orientée produit et d'un point de vue implantation, d'instancier les architectures et les plateformes orientées vers les services liés au futur internet pour des systèmes globaux de production de services.
La thèse à apporter plusieurs résultats (MDSEA, Etended Actigram Star EA*, Transformation de modèle, simulation, et SLMToolBox) pour répondre aux besoins de servitization.

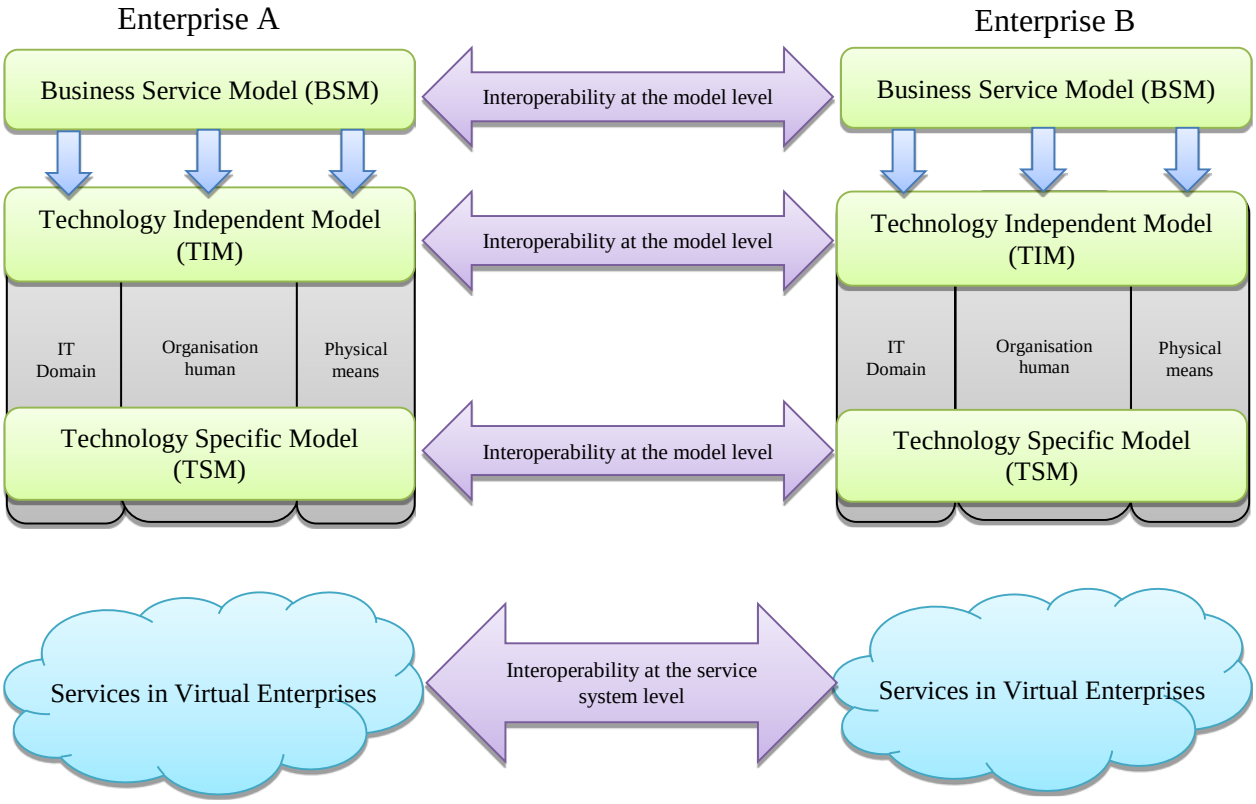


Figure 1 MDSEA

Le MDSEA apporte un cadre méthodologique générique inspiré de l'Ingénierie Dirigée par les Modèles et dont le bénéfice principal est de permettre d'exprimer le traitement de toute question relative au cycle de vie d'un service, au travers de modèles spécifiés à divers niveaux d'abstraction, et reliés entre eux par des mécanismes de transformation de modèle.
Chacun de ces niveaux de modélisation nécessite des langages de modélisation spécifiques. Cette architecture suggère 3 niveaux d'abstraction : (1) un niveau appelé BSM (pour Business Service Model) où sont spécifiés les modèles conceptuels de processus Métier orientés Service à l'aide d'un langage conçu à cet effet, nommé EA* (pour Extended Actigram Star) et inspiré du langage GRAI Extended Actigram ; (2) un niveau appelé TIM (pour Technology Independent Model) où sont spécifiés les modèles détaillés de ces mêmes processus à l'aide du langage BPMN (Business Process Modeling Notation), modèles obtenus par transformation de modèle EA* en BPMN ; et (3) un niveau appelé TSM (pour Technology Specific Model) où sont spécifiés les modèles avec les choix technique spécifique au développement et génération de code. MDSEA ne se limite ainsi pas aux seuls aspects liés aux IT mais aussi aux aspects liés ressources humaines et matérielles devant être prises en compte, créées ou encore achetées pour mettre en œuvre le service attendu et le gérer au long de son cycle de vie.

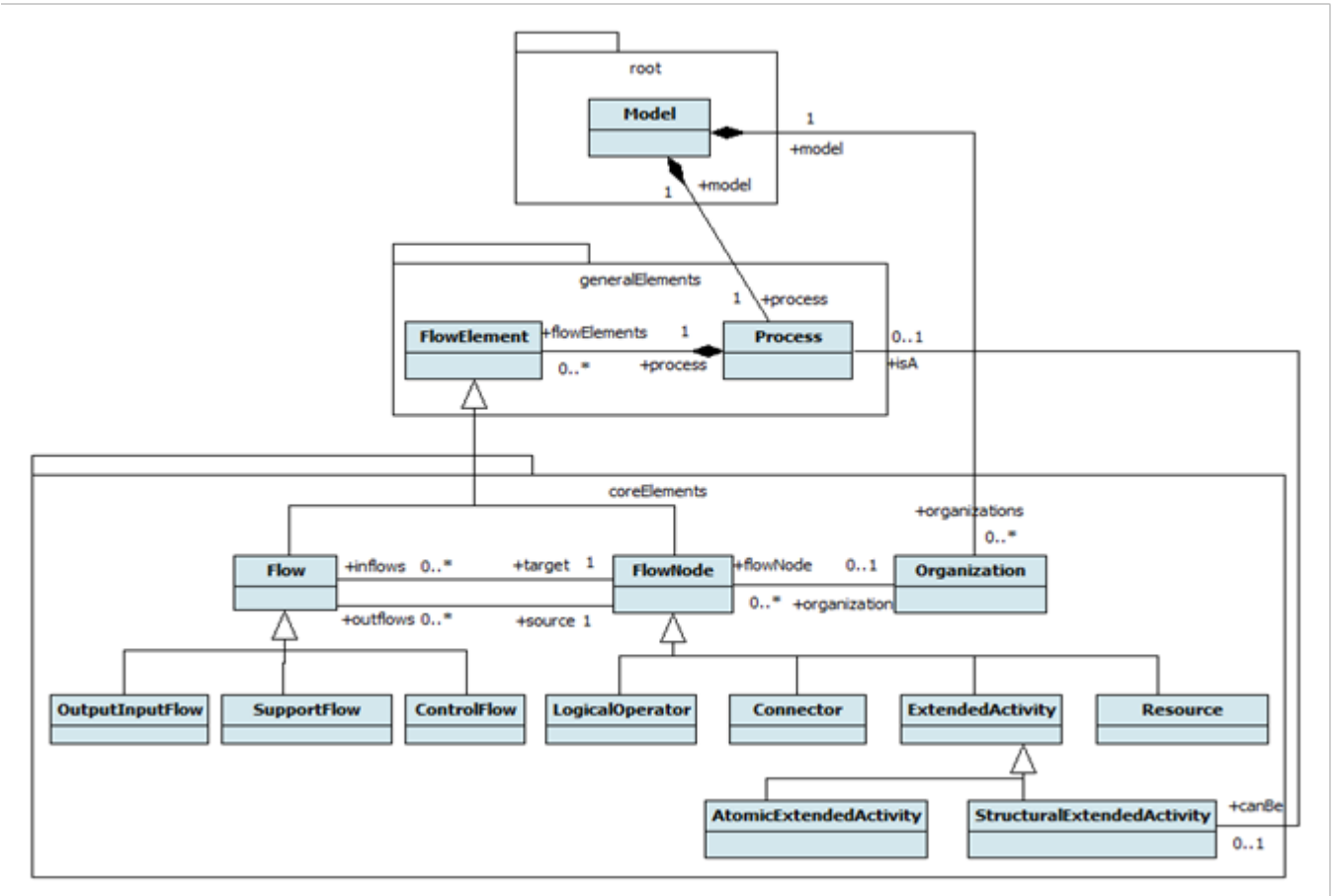


Figure 2 Extended Actigram Star

Extended Actigram Star (EA*) est un langage de modélisation de processus business, développé dans le cadre de cette thèse et inspiré du langage GRAI Extended Actigram. La syntaxe abstraite et concrète de ce langage est décrite de manière détaillée.

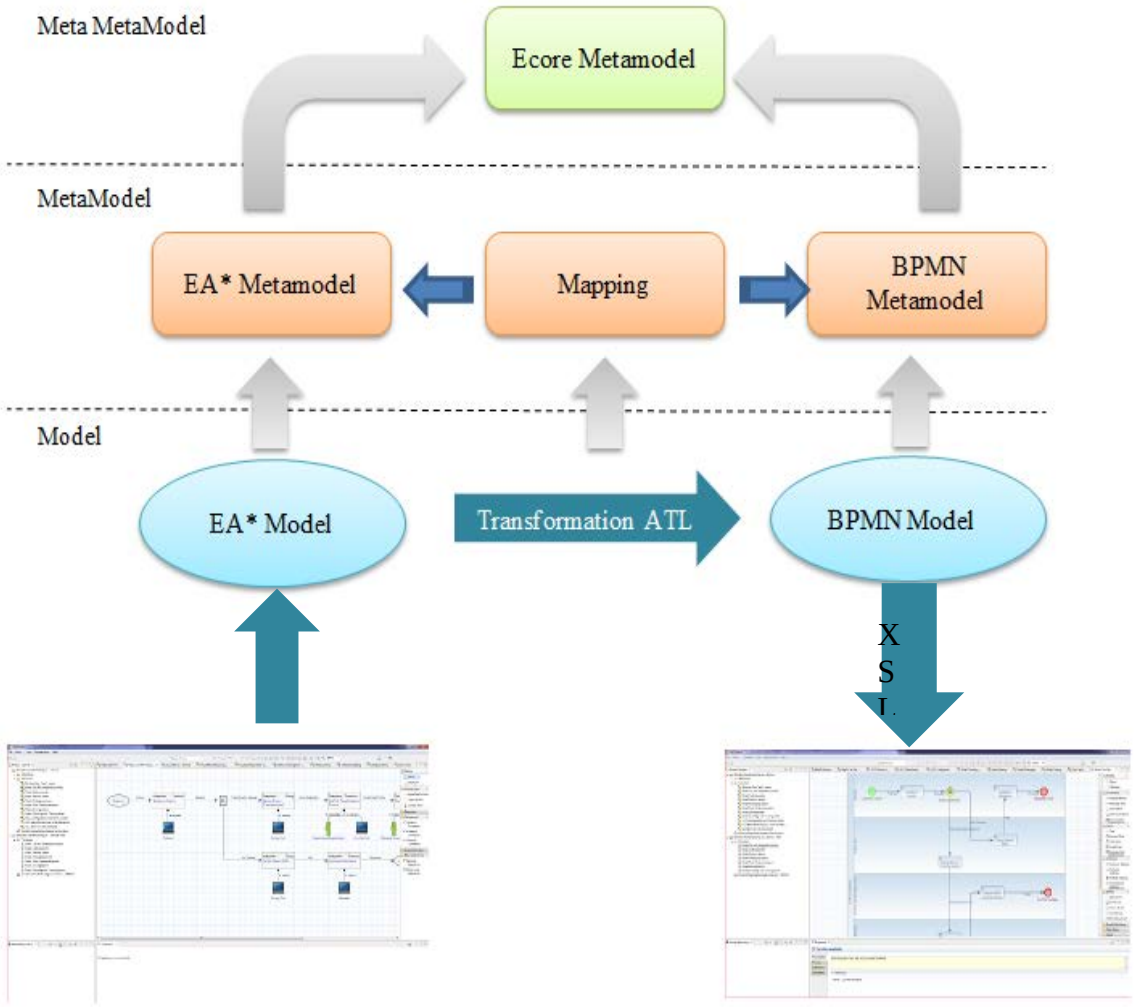


Figure 3 Transformation de model

La transformation de modèle est basée sur le « mapping » entre méta modèles. Le mapping a appliqué est défini après avoir étudié le langage source et langage cible. Puis on a implémenté les règles de transformation en utilisent « ATLAS Transformation Language » ATL. La transformation de modèle nous permet dans le cadre de MDSEA de passer d'un niveau d'abstraction vers un autre. Deux transformations des modelés sont développées pendant la thèse : EA* vers BPMN et BPMN vers DEVS.

La simulation est un outil d'assistance à l'ingénierie, comme au management ou encore au pilotage de systèmes complexes, DEVS est aussi reconnu comme un concept formalisé et largement usité pour la modélisation et la simulation du comportement de système basé sur une hypothèse de comportement à événements discrets. J'ai travaillé sur les règles de transformation de modèle BPMN 2.0 vers DEVS afin d'en permettre la simulation. Cette simulation a pour objet d'évaluer les performances en termes de coût de de temps mais évoque aussi des performances en termes de qualité et de flexibilité. Ces transformations sont à la base d'une sémantique opérationnelle de BPMN i.e. des règles, éventuellement assorties de probabilités d'évolution, décrivant comment un concept de modélisation interagit avec d'autres concepts et évolue en conséquence d'un état au suivant si l'on adopte une vision dynamique basée sur un modèle à états / transitions et événements comme DEVS. Le modèle finale DEVS est simulé au travers des profils de simulation.

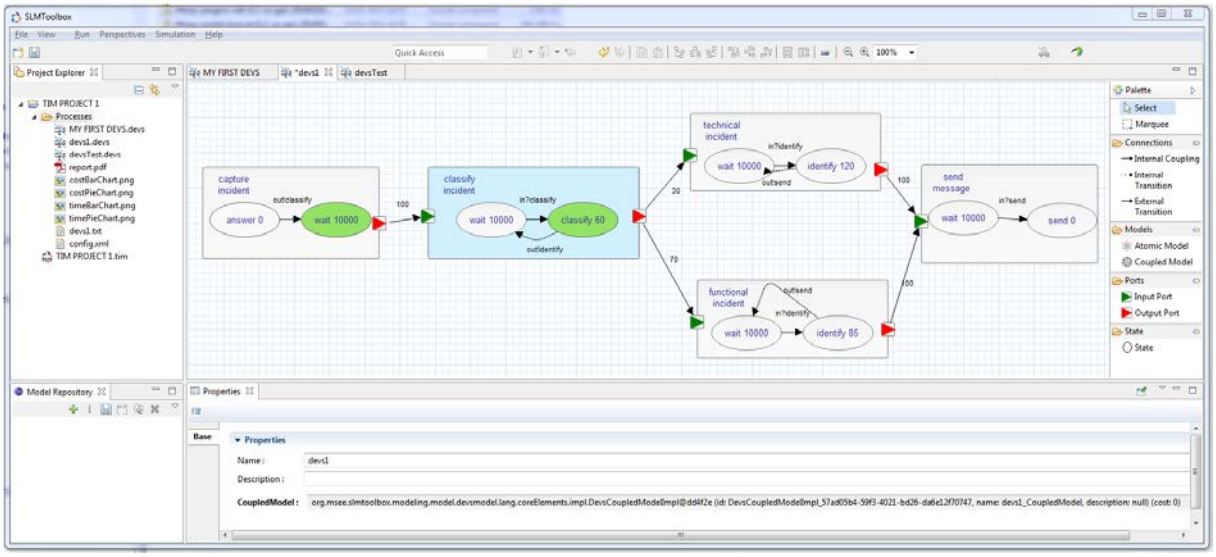


Figure 4 Editeur DEVS

L'outil SLMToolBox intègre les langages adoptés, implémente les règles de transformation de modèle, et offre un environnement de simulation, d'évaluation des performances et d'animation des résultats. Il vient donc en support aux analystes Métier qui, en collaboration avec les experts des différents domaines Métier, peuvent ainsi décrire et évaluer leurs systèmes de services courants (modèles AS-IS), mais aussi concevoir et évaluer de nouveaux services (modèles TO-BE). SLMToolBox est une application Eclipse RCP (Rich Client Platform). C'était développé en java et en utilisent des « Framework » différents : EMF/Ecore (Génération de code et représentation des modèles), EEF (pour gérer les « properties » des objets graphiques), Graphiti (développement des editors graphiques), ATL (Transformation des modèles).

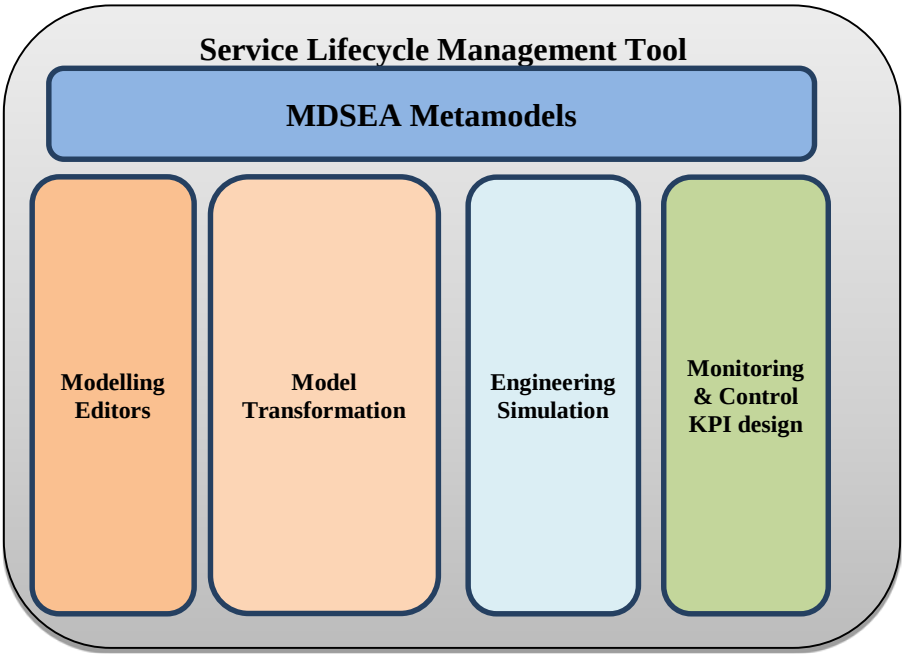


Figure 5 SLMToolBox Architecture Logique

La SLMToolBox est un des résultats apprécié dans le projet MSEE. Plusieurs réunions ont eu lieu à Bruxelles pour la création d'une communauté scientifique autour de la SLMToolBox. Le but de cette Communauté est de reprendre le développement de la SLMToolBox pour l'adapter aux besoins différents des clients dans plusieurs domaines. Grace à cette communauté, le SLMToolBox était utilisé dans plusieurs projets (comme NOSCIFel dans le domaine de transport…).

Mots clés :

Servitization, System de Service, MDSEA, Extended Actigram Star, Transformation de modèle, SLMToolBox, Simulation DEVS.

General Introduction and Problem to Solve: From Service to Service Modeling in Virtual Manufacturing Enterprise Context

1. Context

Around one in ten (9.8 %) of all enterprises in the EU-27's non-financial business economy were classified to manufacturing in 2009, a total of 2.0 million enterprises. The manufacturing sector employed 31 million persons in 2009, generated 5.812 billion Euro of turnover and 1.400 billion Euro of value added. By these measures, manufacturing was the second largest of the NACE sections within the EU-27's non-financial business economy in terms of its contribution to employment (22.8 %) and the largest contributor to non-financial business economy value added, accounting for one quarter (25.0 %) of the total. Furthermore, SMEs were identified as the backbone of manufacturing industry in Europe. Micro, small and medium enterprises provided around 45 % of the value added by manufacturing while they accounted for around 59 % of manufacturing employment [EFFRA, 2013].

In today's world of business, manufacturers are facing many challenges. Business strategies are changing and manufacturers are entering new markets and striving to meet new and changing customer needs. Manufacturers are outsourcing more components and services to suppliers around the world, restructuring their internal operating and information systems, and re-engineering production processes to eliminate waste and lower costs. They are changing the nature of their organizations by partnering with other companies in complex supply chains and business networks that now extend globally. Manufacturing is being redefined by changes in market place and how companies react to them. Some key drivers of change are working across international markets:

- New and more demanding customers
- More demanding stakeholders
- Intense competition
- The pace of innovation and development of new technologies

As a result, many manufacturers wanted to make the shift to services as a solution, but they find themselves trapped in the world of products. Their systems, procedures and practices are all structured to support the design and delivery of products, yet increasingly their customers are demanding services and solutions. A revolution has occurred worldwide in the business of manufacturing. Business is responding to the globalization of industrial markets, production systems, supply networks, and competition. Manufacturers no longer see their activities simply in terms of transforming raw materials into components or finished products. Today manufacturing is a system encompassing all the activities that are required to deliver products that meet customer needs and that extends from research, development, design, and engineering to production, finance, sales, marketing, and after sales service. In simple words, manufacturers have started shifting towards servitization in order to compete in the market.

Servitization, the term coined by [Vandermerewe, 1988], is now widely recognized as the process of creating value by adding services to products. Since the late 1980s its adoption as a competitive manufacturing strategy has been studied by a range of authors [Baines, et al., 2007] [Oliva & Kallenberg, 2003] [Slack, 2005] who have specifically sought to understand the development and implications of this concept. The intuitive understanding of service is gotten by comparing it with the word product. Products are physical entities that are manufactured from raw materials. Services are non-physical entities that are the applications

of knowledge and skills for the benefit of a party [Vargo and Lusch, 2004]. The most famous characteristics of services distinguishing them from products are intangibility, heterogeneity, inseparability and perishability, now known as the IHIPs [Gummesson, 2007].

For the European manufacturing industry of the future, servitization is regarded as one of the most important trends. Innovative combinations of tangible products with intangible knowledge oriented services make the resulting solution more attractive and beneficial for the user and the consumer especially high wages regions which cannot compete on international scales.

2. Principles of Service and Service System modeling

This chapter introduces the principles and concepts of Service and Service System's modeling, development, operation, and governance. For this reason we will start by exploring the main characteristics of Service and Service System in the domain of Manufacturing. Then the **S**ervice **L**ife cycle **M**anagement concept is introduced.

2.1 From service to servitization

Studies and researches in Service's domain have been mostly devoted to support tertiary sector domains (e.g. banking & finance, tourism, trade, public administration), with an obvious focus on ICT. Demand for high customization and growing competition has led to the situation that satisfying customer needs only through tangible products from the core business activities is no longer possible [Johnston et al, 2008]. 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 surrounding products became more and more important in manufacturing. A bundling of physical goods and services is required to augment the complexity on the customer side. Increasing attention has to be given to understand the customer problem and create a suitable solution. Services are added to the tangible product in order to support certain phases of its life-cycle (e.g. call center, customer support etc.). They can be provided by the same company offering the tangible product or by a third party. Furthermore, the services can be included in the price of the physical good or invoiced separately. Therefore, the services have to be closely connected to a tangible product, but they don't have to be necessarily supplied with it.

This evolution is called **Servitization** and its most tangible effect is the development of **Product Service Systems (PSS)**. The process of creating value by adding services to a tangible product has first been called "servitization" by [Vandermerwe and Rada, 1988]. They describe the increasing customer demand-driven offering of product-service "bundles" (consisting of goods, services, support, knowledge and self-service) by modern corporations to create a competitive edge. This requires looking at customer needs as a whole, demanding for new relationships between suppliers and customers. In the beginning, the provision of services has been regarded as a side-show by manufacturing companies. The main value creation was attributed to the tangible product, while services have been added for marketing purposes [Gebauer and Friedli 2005]. However, in many cases services have become as important for the customer as the product itself, so that they are a main differentiating factor for the customer. As a recent development, a larger portion of the added value for the customer is coming from the services, reducing the tangible product to a part of the whole offer [Gebauer et al. 2006]. Different levels of servitization can be identified, reaching from the traditional manufacturer of tangible products, over the provision of service add-ons to the provision of products *as a* service.

2.1.1 **Characterization of a service**

The definition of a Service is very difficult. We extracted from the survey of the literature we performed, one definition we estimate significant: "Service is the application of competence for the benefit of another. Service involves at least two entities, one applying competences 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 a service is difficult to control.
- Service cannot be stored.
- Service delivery lead time is crucial.
- Service delivery integrates physical and electronic way.

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) [Perrey, 2003], 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.

2.1.2 **Product service and PSS**

A product-service is often called a service supplied in addition to a product thus 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:

- **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.
- **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, 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 **user-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 and not for the product. For instance, most of mobile phone providers offer the phone to sell communication time.

An **Industrial Product-Service System (IPS2)** is characterized by an 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 **S**mall **M**anufacturing **E**nterprises (SMEs) or even large industrial companies. In the next section we will analyze the PSS in the domain of Manufacturing and its development process (Servitization).

2.2 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, or new innovate services linked to specific products developed around these services. An appropriate concept to link products, product related services and users' need is the "Extended Product" (EP) (Thoben et al. 2001).

2.2.1 Extended Product (EP)

The Extended Product 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.

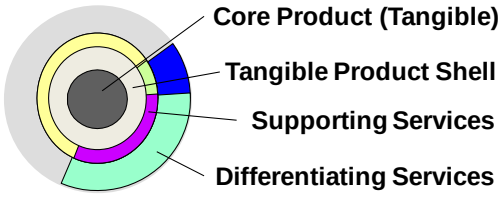


Figure 1 The Extended Product Concept, adopted from (Thoben et al. 2001)

The Core Product is the physical product that is offered to the market; while the Product Shell describes the tangible "packaging" of the product. 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 a 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.

2.2.2 **Product+Service and Product2Service**

The different stages of product and service provision are shown in Figure 2.

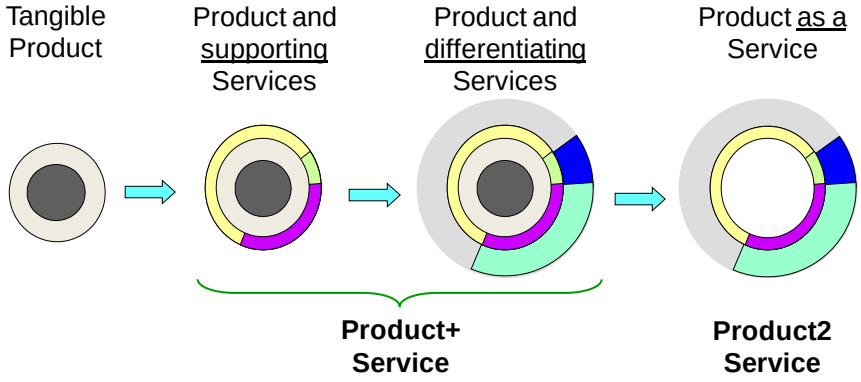


Figure 2 Servitization process

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", starts 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. For example, a washing machine manufacturer will add a device on the machine-tool to check continuously the functioning of the machine). **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 washing machine example, we can propose to sell the machine plus a service which guaranties a high percentage of availability of this machine. Finally, **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. the previous washing machine manufacturer doesn't sell the machine-tools but sells hours of running of the machine-tool).

2.2.3 **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):

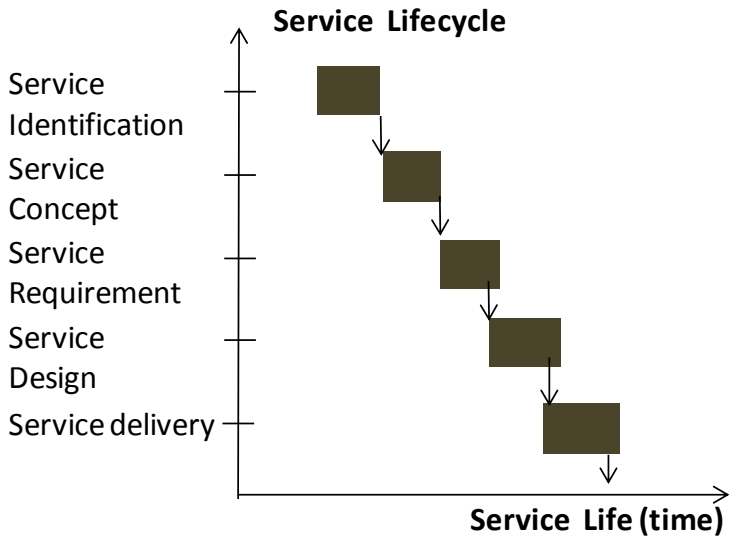


Figure 3 Service Life Cycle

- Service identification: identify service's domain, objectives, and 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.

2.3 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.

2.3.1 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).

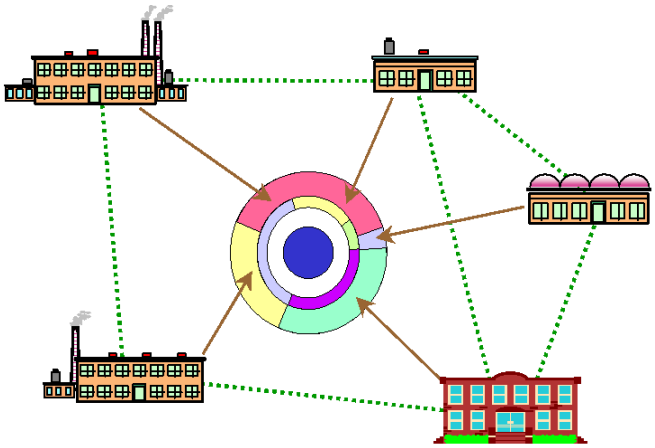


Figure 4 Virtual Manufacturing Enterprise

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.

2.3.2 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 coherence.

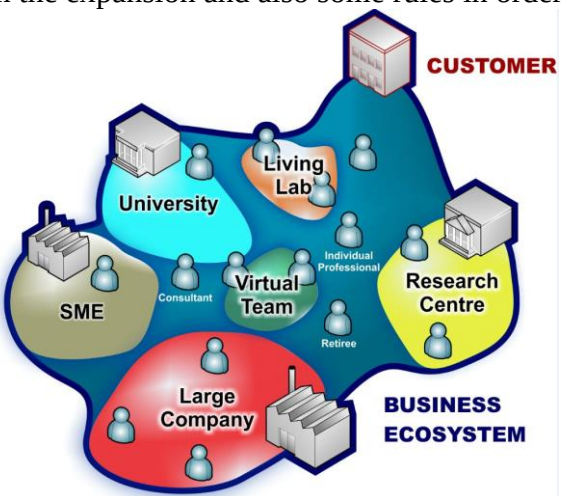


Figure 5 Business Ecosystem Concept

2.4 Service System and Service System Life cycle Management

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

2.4.1 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 Servitization, we reach to 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 (Chesbrough, 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).

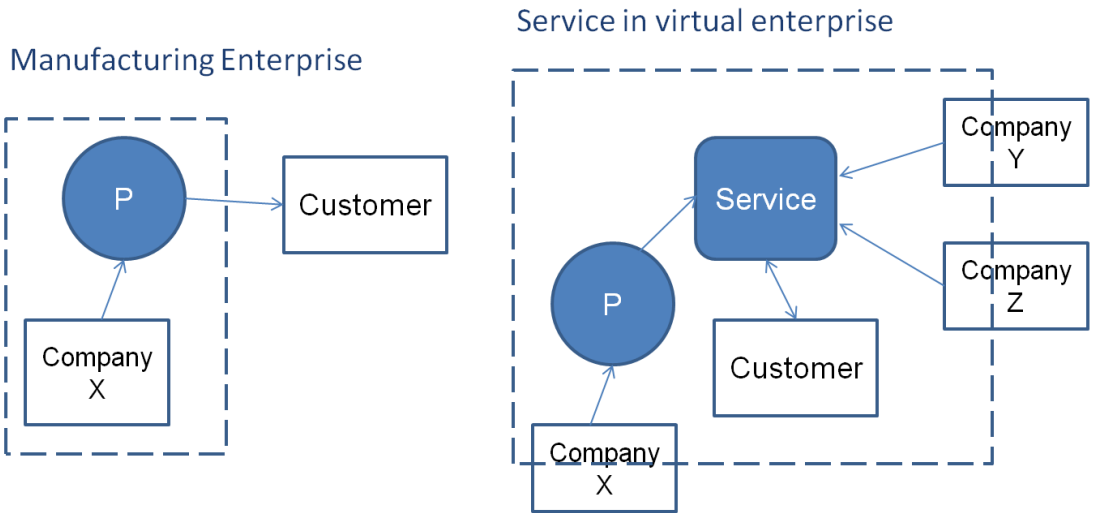


Figure 6 Manufacturing enterprise vs. service in manufacturing virtual enterprise

A **Product System** is composed of an organization which produces a product and delivers a Product to the Customer. This organization 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 organizational form a "Manufacturing Enterprise" and it is represented in figure 6 by a rectangle with a dotted line. There are relations between the "Manufacturing Enterprise" and 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 center 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 addition, 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.

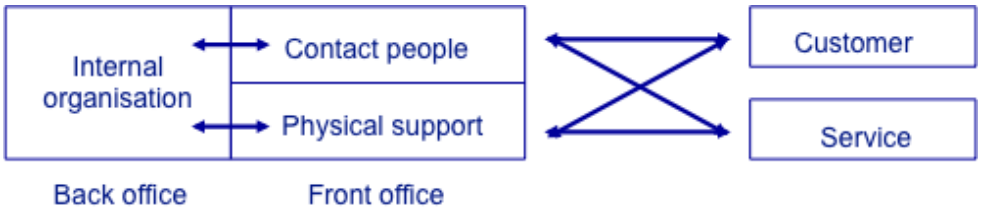


Figure 7 Service delivery system

2.4.2 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 [Williams, 1996], GERAM [Williams, 1995], 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, and 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, and maintenance.
- Service System decommission: end of the system service to remove and destruct it and recycle its components.

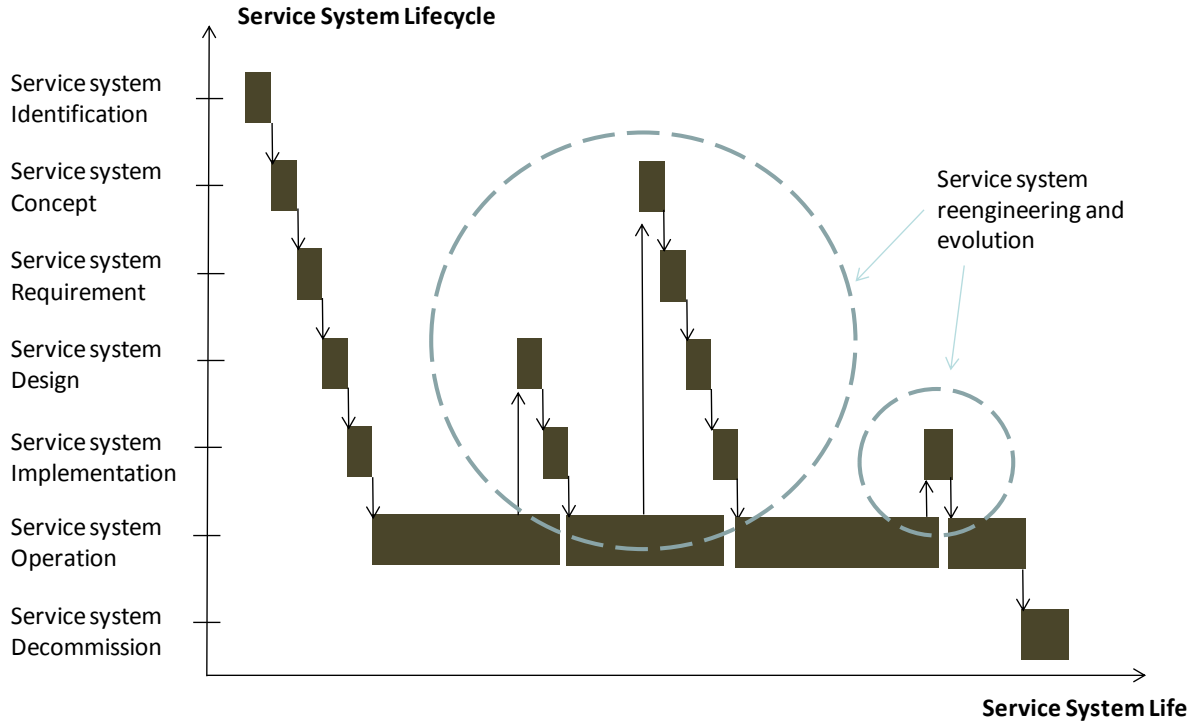


Figure 8 Service System lifecycle phases vs. Service System life - adapted from Bernus (1995)

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.

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 centers will be more involved in the ideation and design phase than in the realization phase.

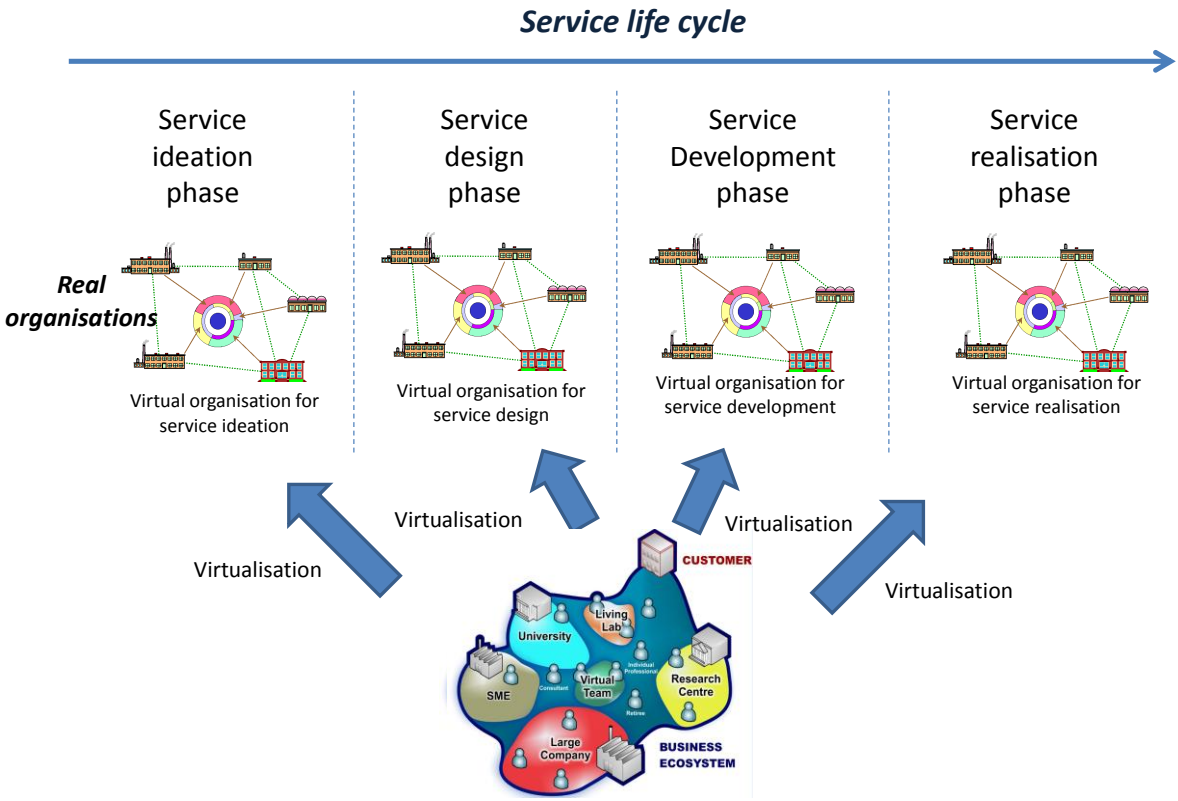


Figure 9 Virtual Manufacturing Enterprise in different phases of SLM

2.4.3 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.
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.
- (2) to (3): Special kinds of services are identified, defined, realized, and these services are recognized by the customer as benefits that differentiate the product from competitors. 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.

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 Service Systems must allow not only the modeling of enterprise's system, but also a system of systems, meaning the VE or even the MSE which generated it. In such a case, it will be necessary, by comparing AS IS and TO BE models at each stage, to determine the evolution of the Service System, and in particularly the:

- IT system
- Organization and Humans supporting 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.

2.5 Modeling of Service System

2.5.1 Why to model 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 centers 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 as 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 which should be analyzed carefully. We propose to be inspired for 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.

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 Enterprise Modelling and by extension in Service System. In Enterprise modelling it is necessary to consider two views: a global view which allows capturing the global structure in order to understand the objectives and a local view for modeling detailed elements but in a coherent way with the global view.

Herbert Simon, one of the founder of 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).

2.5.2 **Modeling Service System using System Theory**

In this section we will introduce the system theory as being the starting point for building service systems or system of systems.

2.5.2.1 System Theory

The system theory is the result of the research works done by many authors among whose we can refer to Herbert Simon, Jean-Louis 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, organization, 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.

- 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.
- 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.
- 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.
- **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 also allow the definition of the system's borders. 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.

- **Finally a system is dynamic**, which means that it evolves according to 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 various possible status of a system. So, a system can be represented by the Figure 10 below:

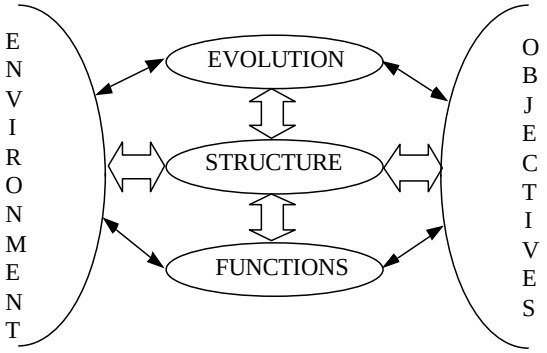


Figure 10 The structure of a system

These concepts are a valuable reference for the modelling of one enterprise. But in the frame of servitization and 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 as 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 Figure 11. However, even if this is difficult to represent, figure 11 must indicate that between the systems there is more than common objectives. Otherwise, the complete organization is not a system. So, the system of system must integrate common functions, structure and evolution. **Based on this definition, the system theory aims to represent (to model) the realities of a system, concretes or abstract, highlighting at the same time global and detailed representation of this system.**

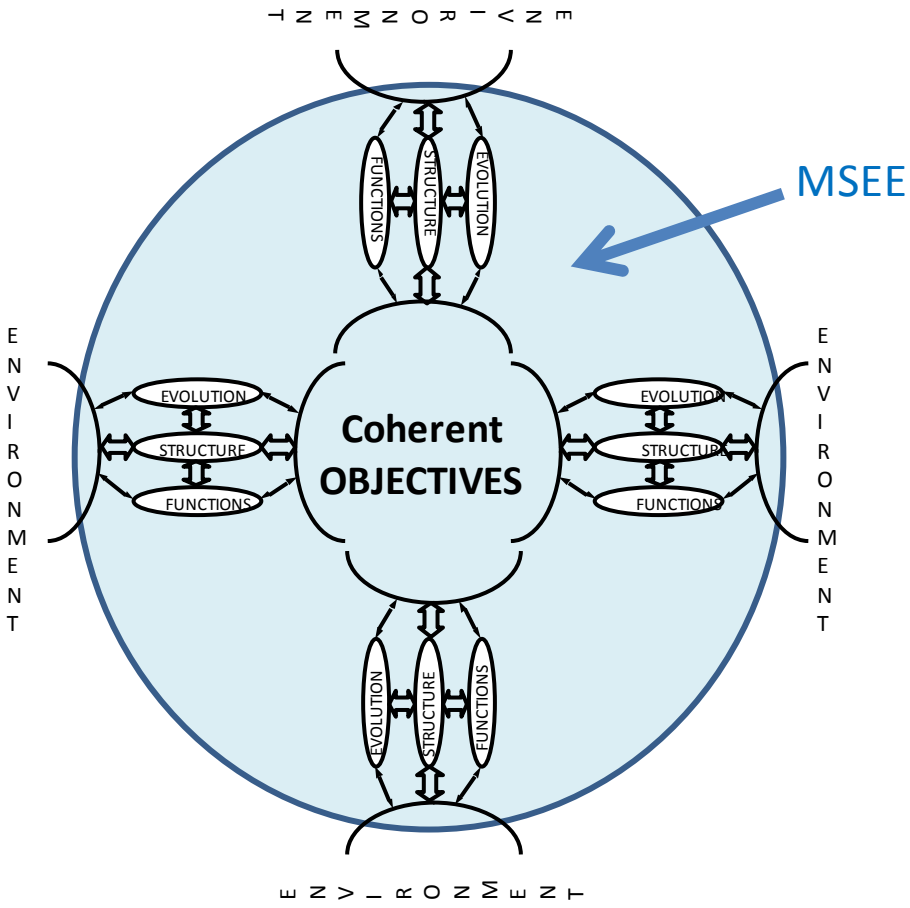
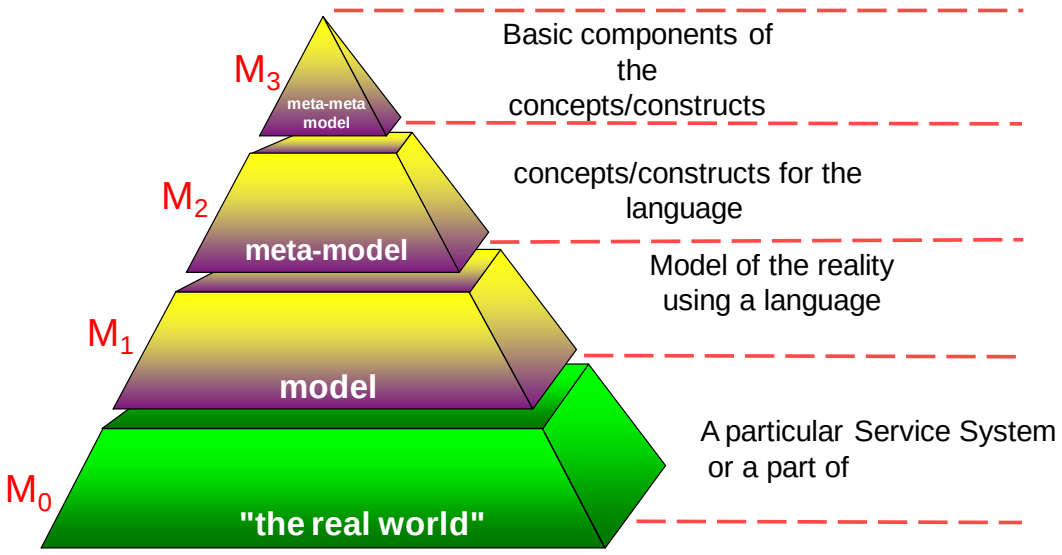


Figure 11 MSEE as a system of systems

2.5.3 Definition of the languages to describe and represent the Models

2.5.3.1 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 Figure 12. At level M0, we find the real world; it means all the concrete components which allow building 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 building the language. At level M3, the basic components of concept/constructs are defined.



[Adapted from OMG, Bézivin, Aber Wrach /16-20 Septembre 2002]

Figure 12 Enterprise Modelling mapped to the OMG 4 level architecture

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 (virtualization = abstraction M1)

2.5.3.2 Modelling language concepts and constructs

At the level M2, (Figure 12) 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 adopted approach 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 level M1, we find the models described based on the languages chosen at level M2.

2.6 Architecture for Service System Engineering

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

2.6.1 **Service system engineering**

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, **Organization/Human** domain (including management and organization), 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. Manufacturing Service Ecosystem (MSEE) Project

MSEE is an Information and Communication Technology (ICT) integrated project, funded by the European Commission in call Factories of the Future (FoF) of the 7th Framework Program. The general objective "Virtual Factories and Enterprises" focuses on end-to-end integrated ICT solutions that enable innovation and higher management efficiency in networked enterprise operations. The MSEE project promotes new concepts, methods, and tools for innovative collaborative services between various partners.

The MSEE 2015 Vision stems upon two complementary pillars, which have characterized the last 10 years of research about Virtual Organizations, Factories and Enterprises: Service Oriented Architectures (SOA) and Digital Business Ecosystems (DBE).

The first Grand Challenge for MSEE project is to make SSME (Service Science, Management and Engineering) evolve towards Manufacturing Systems and Factories of the Future from:

- Methodological viewpoint to adapt, modify, extend SSME concepts so that they could be applicable to traditionally product-oriented enterprises;
- Implementation viewpoint to instantiate Future Internet service oriented architectures and platforms for global manufacturing service systems.

The second Grand Challenge for MSEE project is to transform current manufacturing hierarchical supply chains into manufacturing open ecosystems:

- Define and implement business processes and policies to support collaborative innovation in a secure industrial environment;
- Define a new collaborative architecture for ESA, to support business-IT interaction and distributed decision making in virtual factories and enterprises.

3.1 MSEE Results

The MSEE project produced several interesting scientific/technical outcomes which are duly described in the rest of this book. As an executive summary, we could classify them in three broad categories: MSEE generic assets, MSE specific assets, and VME specific assets.

3.1.1 **MSEE Generic Assets**

MSEE Generic Assets are to be used by manufacturing enterprises to improve the relevance and role of services in their business:

- Maturity, Positioning and Change Management: manufacturing enterprises are able to approach the MSEE world by gradually understanding their maturity levels with respect to service innovation and collaboration. The most suitable intervention areas and MSEE assets are identified and proposed in a game-like approach.
- Service Strategy and Business Models: a service innovation strategy and relevant new Business Models need to be carefully analyzed and evaluated before putting them in operations. MSEE offers a suite of methods and tools to drive manufacturing enterprises towards a more mature, aware and engineered servitization of their business.
- Reference Architecture for ESA: MSEE reference architecture is an evolution of the Service Delivery Platform aiming at defining three main levels for FI inspired Enterprise Systems (the level of the single enterprise, the level of business ecosystems and the level of the Internet of the Future) as well as two main alignment-interoperability flows, the former for models-knowledge, the latter for services-platforms.
- SLM Integration Platform: this 3rd generation platform includes and integrates platforms, applications and services along the Service Lifecycle, such as the Service Ideation, Service Modelling, Service Development, Service Delivery, Service Mobile platforms and the value added services for Service Operations (e.g. IoT Manager, Marketplace, Team Building, Feedback Management, Production Planning, Product Maintenance). A generic platform for Business Intelligence has been also recently integrated. The four SLM Platforms instantiated in our test cases are also including additional applications and assets necessary for the integration of their business processes.

3.1.2 **MSE-Specific Assets**

MSE-Specific Assets, which aim to improve the collaboration along the product-service lifecycle, by setting-up, managing and governing a Manufacturing Service Ecosystem (MSE):

- MSE Management and Governance: an MSEE MSE is a non-hierarchical collaboration form whose organizational structure, decisional processes and management procedures are flexible, dynamic, in some cases non-deterministic, to allow the necessary agility required by service innovation.
- Service Ideation in MSE: the final aim for the existence of an MSEE MSE is to create an incubation and acceleration environment for new ideas of services, to be easily

evolved into concrete assets.

- Virtualization of MSE Tangibles / Intangibles: the heterogeneity of the resources of an MSEE MSE should be dominated by common unified representation of such diverse artefacts, i.e. virtualization and representation as a service, so that they could be used and exploited by service marketplaces.
- MSE IT Platform (IEP): as the open source one-stop shop for all the members of an MSEE MSE , able to offer IT support to MSE operational business processes governance, collaborative service ideation models and management of virtualized representations of the MSE assets.

3.1.3 **VME-Specific Assets**

VME-Specific Assets aims to improve the service engineering maturity, by setting- up, managing and governing service-driven Virtual Manufacturing Enterprises (VME):

- Servitization Framework for VMEs: manufacturing companies willing to pursue a servitization project via a VME collaboration need to identify and select the most suitable strategy as well as the most proper methods, tools and IT. The MSEE servitization framework, including role- and competency-based models, provides MSEE with models and tools to select the best partners for servitization and set-up efficient and effective VMEs.
- VME-oriented Service Life Cycle: the implementation of a Service along its lifecycle (i.e. design, development, testing, deployment, operations' dismission) implies the constitution of several different VMEs linked together by a common Service Lifecycle Management model, which, in the case of manufacturing companies, needs to be integrated from organizational and temporal viewpoints with the pre-existing Product Lifecycle Management model.
- Service Modelling Architecture for VME (MDSEA - Model Driven Service Engineering Architecture): this reference architecture allows VMEs to model and refine their service design and development processes through several different abstraction levels, starting from the business perspective and proceeding top down along three main action lines, namely Organization, Physical Means and IT. Business criteria and indicators are also modelled and accompany the service models in their refinements and transformations.
- Service Modelling Toolbox (previously SLM Toolbox): this open source IT component encompasses editors, knowledge and model bases as well as KPIs repositories, following the top-down decomposition of a service and its implementing system.

4. Contribution of the thesis

Based on the problematic of Servitization and service system engineering and in order to reduce effort and time in service system engineering, this thesis (as being part of the MSEE project) contributed in the development of solutions. The contribution of the thesis's result can be classified into related and connected pillars.

The first pillar is the participation in the development of the Model Driven Service Engineering Architecture (**MDSEA**) which permits Virtual Manufacturing Enterprises (VME) to model their service systems (AS-IS and TO-BE models) starting from modeling the system

from business experts angle and then adding more details to reach the developers and technical experts angle. We propose the principles, concepts, and languages for Service System's modeling which will not only generate IT applications and services but also define other components (Organization/human and Physical Means) which will support all Service Life Cycle phases. The method models the service system from several views and using specific standard and non-standard modeling languages. One of these languages, the **Extended Actigram Star (EA*)** was redeveloped in the frame of this thesis based on an older language, the GRAI Extended Actigram. The EA* is used in the MDSEA methodology as a collaborative business process modeling language at high abstraction levels. In addition, **model transformations** are specified, developed, and implemented. Since MDSEA is composed of several abstraction levels, model transformations are needed in order to move from one level to another.

The second pillar is the development of a modeling and simulation tool, the **SLMToolBox**. This tool is a partial implementation of MDSEA and its name Service Lifecycle Management ToolBox implies a role in the service's lifecycle. The SLMToolBox offers several features for its different actors. Various domain specific graphical editors are developed or integrated which gives the opportunity to users to model diagrams corresponding to a specific modeling language to be used in the MDSEA methodology. Also, Performance indicators can be modeled and added to specific diagrams. The different features are to be detailed in a single chapter.

The third pillar is the development of a **DEVS graphical editor and simulator** integrated in the SLMToolBox. DEVS diagrams can be either developed from scratch or the result of a transformation from **BPMN diagrams to DEVS diagrams**. After developing the diagram and inserting the two performance indicators to simulate (time and cost), simulation can be executed. The results are in the form of a pdf report and animations of these results can be executed also.

5. Organization

Chapter 2 is a state of the art of principal concepts which constitute the base our research work strongly depended. Enterprise modeling and interoperability are explored in order to provide an insight on the work in previous projects and researches. Then Model driven Developments such as MDA and MDI are presented which will later represent a basic inspiration in our work. After that three modeling languages are sighted: GRAI Extend Actigram, BPMN, and DEVS. Both GRAI Extended Actigram and BPMN being related to business process modeling, while DEVS representing simulation formalism to study the system's behavior. Finally, a set of business processes and DEVS simulation tools are sighted giving a briefing of major tools available and their different criteria.

Chapter 3 is our contribution to service modeling. It presents the Model Driven Service Engineering Architecture (MDSEA) which is targeted to the representation of service systems and the management of certain aspects in the service's lifecycle. Besides, we propose the Extended Actigram Star (EA*) process modeling language, that we specified and developed based on the GRAI Extended Actigram language.

Chapter 4 is related to service engineering and in specific service simulation. In this chapter we explain the importance of simulating service's behavior. In addition we present a DEVS editor/simulator we developed and which is targeted to simulate business processes in service systems. Details on simulation's execution are detailed.

Chapter 5 is a representation of the SLMToolBox that is regarded as one of our main contributions. In this chapter we present the SLMToolBox, a modeling and simulation tool developed as a partial implementation of MDSEA. Context, objectives, architecture, features, and modeling editors are explained.

Chapter 6 is a general conclusion and defines future perspectives to be conducted in future work. These perspectives are based on ideas for evolving certain features of the SLMToolBox.

State of the Art

This chapter is a lecture on basic concepts and subjects that influenced our research work and formed a base to develop new ideas, methods, and tools. The topics in this chapter are the following: enterprise modeling, enterprise interoperability, model driven development, modeling languages, and simulation tools. This chapter will present the Model Driven Service Engineering Architecture (MDSEA) and Extended Actigram Star (EA) developed in the frame of this thesis.*

1. Enterprise Modelling

Enterprises operating in most industrial and service sectors face a number of business challenges that exceed the scope of the daily operations and routine activities. Examples are continuous process improvements for increased efficiency, adjustments of the enterprise strategy to new market demands, changing business models due to new competition, new regulations and bylaws requiring operational changes, or technological innovations leading to changes customer behavior and new processes. In many cases improving business process alone is not sufficient for addressing problems of this nature. The overall situation of the enterprise has to be taken into account including relations between strategic goals, business rules, work process organization structures, products, services, It infrastructure, etc.

Enterprise Modeling addresses these kinds of challenges. The area of enterprise modeling in general is concerned with techniques, methods, and tools for modeling organizations and for finding and preparing potential improvements. This section is a lecture of basic enterprise modeling approaches and architectures that resulted from continuous research and industrial activities.

1.1 CIMOSA

CIMOSA "Computer Integrated Manufacturing Open System Architecture" [Zelm et al, 1995] [Vlietstra, 1996] is an enterprise modelling framework which aims to support the enterprise integration of machines, computers and people. The framework is based on the system life cycle concept, and offers a modelling language, methodology and supporting technology to support these goals. CIMOSA generic building blocks and modelling macros support model engineering through business users rather than IT professionals. CIMOSA is based on a process oriented modelling approach describing all enterprise activities in a common way. Such activities include manufacturing processes on the shop floor, as well as management and administrative processes. CIMOSA modelling covers the life cycle phases of operational system from business requirements definition to system implementation description, operation and model maintenance even enabling model based operation control and monitoring.

1.1.1 **Approach**

CIMOSA provides a framework for guiding Computer Integrated Manufacturing (CIM) users in enterprise system's design and implementation, and CIM vendors in system component development. It provides a descriptive methodology supporting the System Life Cycle. CIMOSA does not provide a standard architecture to be used by the whole manufacturing industry, but rather a Reference Architecture from which Particular Architectures can be derived which fulfil the needs of particular enterprises. The Reference Architecture provides constructs for structured description of business requirements and for C1M system design and implementations. CIMOSA compliant enterprise systems support organizational and operational flexibility, extensive use of multi-disciplinary enterprise information (knowledge) and graceful system integration. Through the business modelling framework a generic modelling concept is provided which is applicable to enterprises in many industries. Model

execution in heterogeneous manufacturing and IT environments is supported by the implementation of an integrating infrastructure. CIMOSA supports new paradigms in enterprise management enabling explicit description of enterprise processes at different levels of abstraction for strategic, tactical and operational decision support. Applying CIMOSA modelling methodology should results in complete descriptions of enterprise domains and their contained Business Processes including relationships to external agencies (suppliers, customers, even government regulatory bodies, etc..). This enterprise model is stored on and manipulated by the relevant information technology base of the enterprise. CIMOSA allows modelling of the enterprise to be done incrementally rather than following an overall top-down approach. It structures the enterprise operation into a set of interoperating Domain Processes exchanging results and requests. Different views of the manufacturing enterprise content and structure are required to satisfy the needs of the different users of such architecture. CIMOSA provides the necessary constructs to enable these multiple views to be created and manipulated by those users who have specialist knowledge of their particular field but are not experts in IT.

1.1.2 **Overview**

To satisfy the above issues of Management of Change, Flexibility and Enterprise Integration CIMOSA provides three inter-related concepts:

- Modelling Framework (Reference Architecture, Particular Architecture, and Enterprise Model).
- System Life Cycle and Environments (Engineering and Operation).
- Integrating Infrastructure.

CIMOSA recognizes previous efforts in enterprise integration especially in the manufacturing industry and draws from the experienced gained in enterprise modelling and computer systems integration.

1.1.2.1 Modelling Framework

CIMOSA Modelling Framework provides guidance to enable end users to model enterprises and its associated CIM system. CIMOSA modelling approach is based on a Reference Architecture from which Particular Architectures and Enterprise Models can be developed. The structuring and decoupling of user concerns from implementation constraints provided by the framework contributes to enterprise flexibility. The Modelling Framework provides a structure which clarifies relations between parts that make up the enterprise operational system (Information Technology and Manufacturing Technology Components) and methods and software tools that are required to describe, simulate and operate such industrial system.

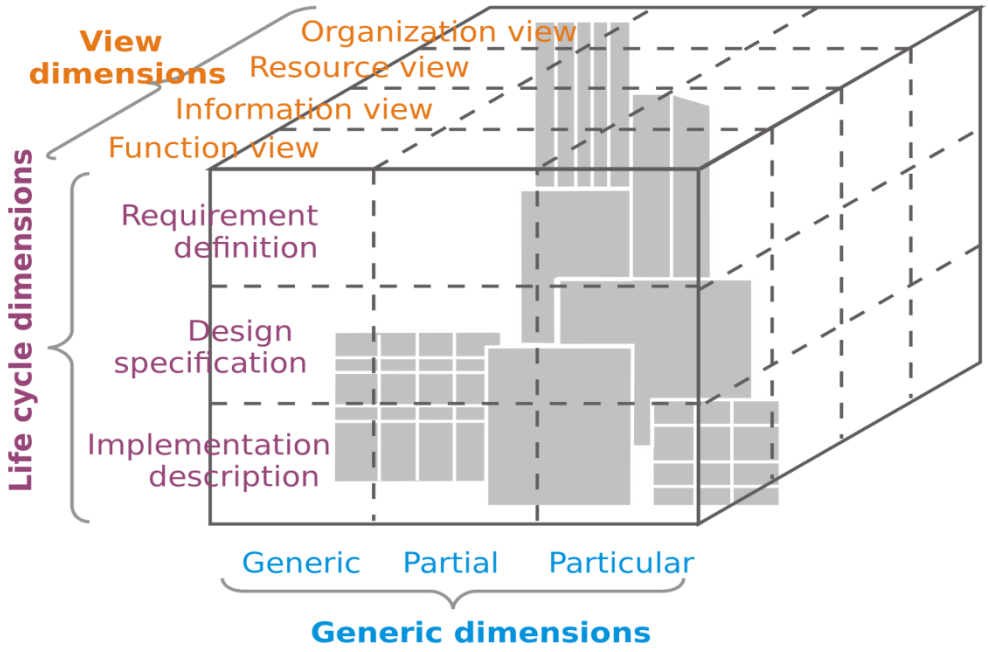


Figure 13 CIMOSA Modelling Approach

When modelling an enterprise there are many aspects and viewpoints to be examined that cannot be structured in one dimensional framework. CIMOSA identifies a three-dimensional framework offering the ability to model different aspects and views of an enterprise:

- **Genericity dimension** concerned with the degree of particularization. It goes from genuine building blocks to their aggregation into a model of a specific enterprise domain. This dimension differentiates between Reference and Particular Architecture.
- **Modelling dimension** provides the modelling support for the System Life Cycle starting from statements of requirements to a description of the system implementation.
- **View dimension** concerned with system behavior and functionality. This dimension offers the user to work with sub-models representing different aspects of the enterprise (function, information, resource, organization).

1.1.2.2 Enterprise Model

According to the structure provided for the particular architecture, CIMOSA models capture business knowledge in terms of

- Domain Processes and Enterprise Activities representing detailed local functionality.
- Business Processes representing intra process behavior.

All required inputs and produced outputs (information, control, resources and organizational) are identified. Modelling is done through instantiation of generic building blocks and partial models. Inside an enterprise, tasks (Domain Processes) are organized into sub-tasks (Business Processes, Enterprise Activities, Functional Operations) which need to be realized to achieve business objectives. Domain Processes are triggered through requests or events and are capable of exchanging information with domains external to the enterprise. In order to represent tasks and actions performed within an enterprise, CIMOSA offers the terms "processes", "activities" and "operations", where operations define the lowest level of

granularity. The level of detail to be described in the model is at user's discretion and not dictated by CIMOSA.

CIMOSA differentiates between AS-IS and TO-BE modelling. Modelling of an existing implementation will start with bottom up description of the current operation. Abstraction of AS-IS description and applying modifications would lead to the specification of TO-BE model's requirements and design and analysis of its intended behavior. No specific methodology has been prescribed by CIMOSA leaving freedom for iterations as required between decomposition and aggregation as well as between modelling levels.

1.2 GIM

GIM (GRAI Integrated Modeling) [Chen et al, 1996] belongs to the set of methods that form the GRAI methodology whose principle objective is performance improvement of both industrial and service enterprises. GIM is based on the GRAI Methodology, particularly on the GRAI conceptual model, formalisms that translate the GRAI conceptual model and GRAI general approach. GIM is the module of the GRAI methodology that allows modelling an enterprise or a part of it in order to improve performance. Based on the GRAI general approach, figure 14 represents the GIM approach and its different phases.

1.2.1 GIM phases

1.2.1.1 Initialization phase

This phase allows to prepare the study and to develop a detailed program to be approved by specialists. GRAI specialist will lead the study and may be assisted by one or more assistant. In this initialization phase, a group is specified to control the objectives of the study and the delineation of the area under study. The study is prepared by establishing a very specific schedule. GIM study includes meetings and interviews with decision makers.

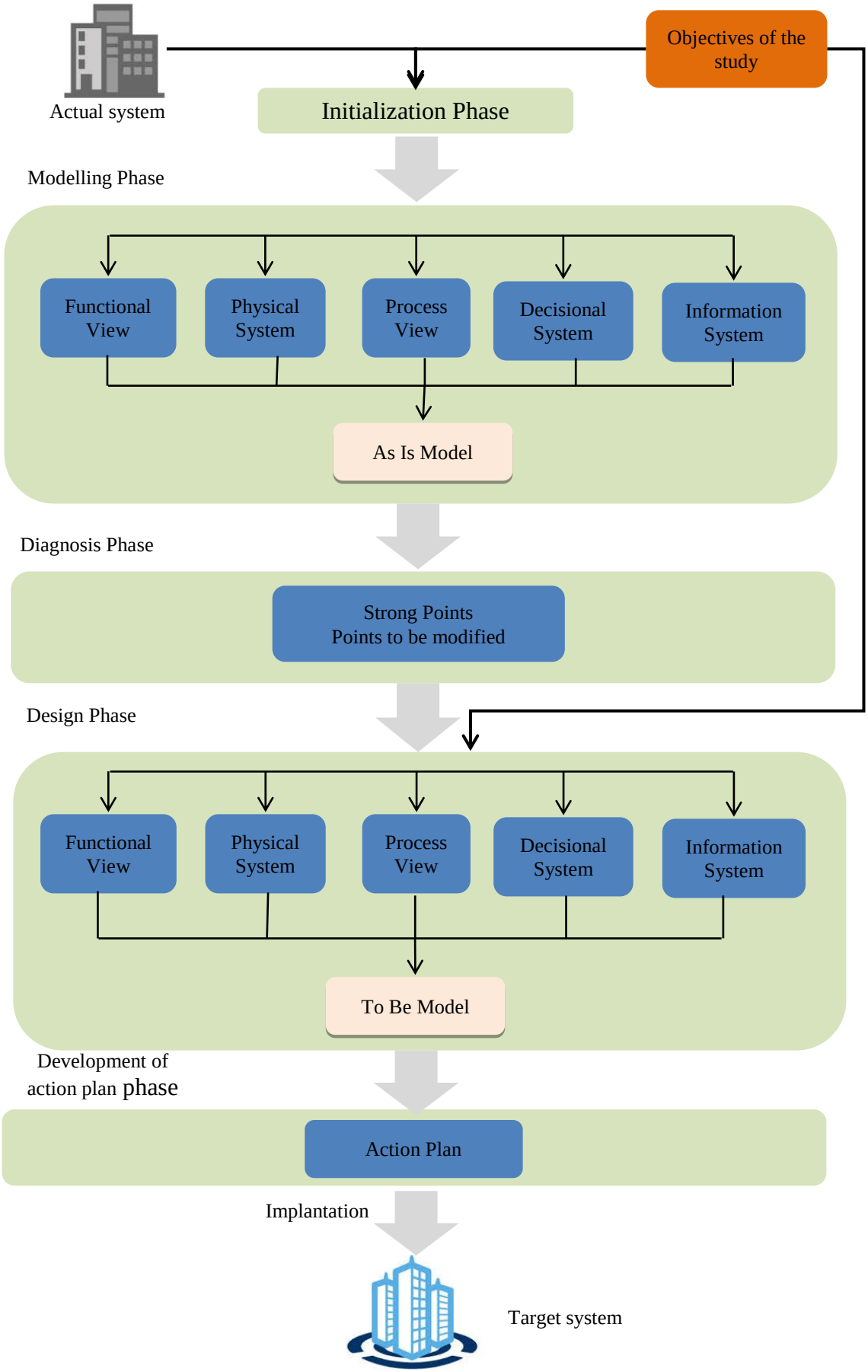


Figure 14 GIM approach

1.2.1.2 Modeling phase

This phase allows modeling of existing system. It includes the realization of the functional view, the three systems (physical, decision making, informational) and the process view. The order in which the model is developed is significant (Figure 15). It starts with the Functional View, and then Physical System and Decisional System are realized in parallel. At the end, the process view is realized and the model of information system is derived from the various realized modules.

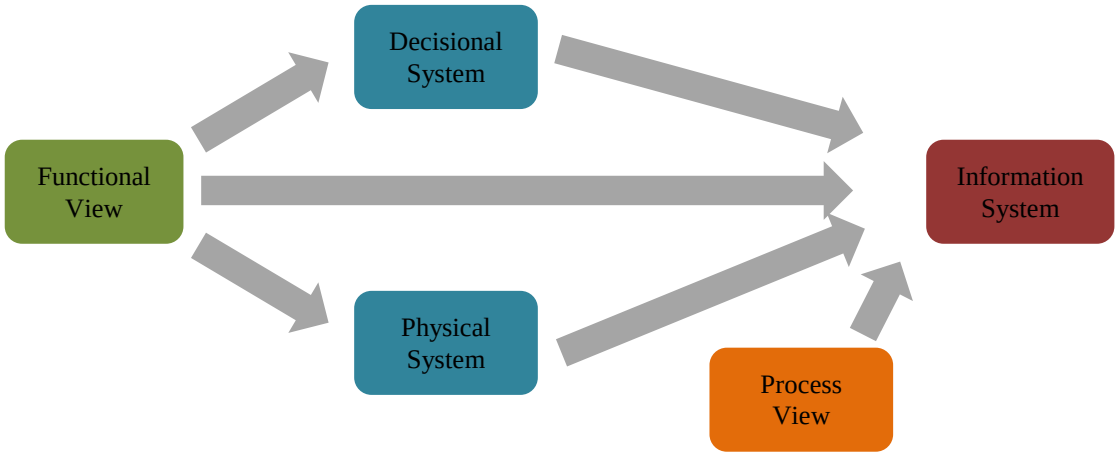


Figure 15 Order of models realization

1.2.1.3 Diagnosis phase

Models diagnosis aims to detect not only areas to be improved but also strength ones. The previous phase has already led to improvements. At the end of each meeting, it is possible to compare the models obtained with the synthetic group (top-down) and information collected after interviews. Sometimes models do not match; the synthesis group had a point of view while the terrain modeling shows that the situation is different. Presenting models of the existing system to the synthesis group is of objective to propose improvements based on experience, knowledge of the field of study and discussion among the members of the synthetic group. Final models obtained at the end of the modeling phase have already helped to make improvements to the existing system which also may in some cases be implemented immediately without waiting for the end of the study. It is possible to say that the final version of the models describe a stabilized situation. Nevertheless, a summary of all areas to be improved is delivered.

1.2.1.4 Design phase

The objective of this phase is to develop models of the target system, models that respond to the objectives defined by the study and can meet the "areas to be improved" identified in the previous phase while retaining the strengths identified. Several solutions are proposed in this phase and only one comprehensive solution should be chosen based on an assessment of the objectives.

1.2.1.5 Development of action plan phase

The implementation of GRAI methodology corresponds to an objective that has been defined by the General Management. The action plan depends on objectives of the study and the “to be” model developed in the design phase. The action plan is later implemented in the system in order to obtain the target system which is the subject of the whole study.

1.3 ARIS

ARIS (Architecture for Integrated Information Systems) developed by A. W. Scheer, is both a generic modelling framework generic and a modelling tool of business processes [Scheer, 1993] [Scheer, 2002] [Scheer et al., 1994]. It mainly focuses on software engineering and on aspects organizational design for integrated systems within the company. Modelling of business processes and all relating factors and domains is seen as a critical and decisive competitive factor by ARIS. The objective of ARIS is to define standardized general concepts (so-called architectures) for IT systems and modelling methods development.

1.3.1 Concept of ARIS architecture

The design of ARIS is a based on an integration concept which is derived from the analysis of business processes. The ARIS framework is structured in terms of five different views (organization, data, control, function and output) and three abstraction layers (Requirements definition, design specification and implementation description.). The purpose is to ensure a consistent description from business management-related problems all the way down to their technical implementation.

1.3.1.1 Views

Organization view
The organization view presents the hierarchical organization structure. It is created in order to group responsible entities or devices executing the same work object. This is why the responsible entities "human output", responsible devices, "financial resources" and "computer hardware" are allocated to the organization view.

Data view
The data view comprises the data processing environment as well as the messages triggering functions or being triggered by functions. Preliminary details on the function of information systems as data media can be allocated to data names. Information services objects are also implicitly captured in the data view. However, they are primarily defined in the output view.

Control view/Process view
This view displays the respective classes with their view-internal relationships. Relationships among the views as well as the entire business process are documented in the control or process view, creating a framework for the systematic inspection of all bilateral relationships of the views and the complete process description.

Function view
The processes transforming input into output are grouped in the function view. The designations "function", "process" and "activity" are used synonymously. Due to the fact that functions support objectives, yet are controlled by them as well, objectives are also allocated to the function view – because of the close linkage. In application software, computer-aided processing rules of a function are defined. Thus, application software is closely aligned with "functions", and is also allocated to the function view.

Output views
The output view contains all physical and non-physical input and output, including funds flows.

1.4 Conclusion on Enterprise Modeling

There is no explicit consideration on interoperability issues in CIMOSA modelling

framework. However, CIMOSA can be a contribution for integrated paradigm to establish interoperability. The CIMOSA Framework is a good reference framework, but lacks expressiveness for multiple dependencies of types of view, for evolving concepts, contents and capabilities and for capturing context. Knowledge sharing and representation is poorly supported. On the other hand GIM modelling framework introduces the decision dimension/view which is not taken into account in other modelling frameworks. The decisional aspect is important to establish interoperability in the context of collaborative enterprises. To interoperate in such an environment, decision-making structure, procedure, rules and constraints need to be clearly defined and modelled so that decentralized and distributed decision-making can be performed. The GRAI Framework has strong support for performance indicator management and decision making, but has limited scope and expressiveness and lacks platform integration. On the other side the different views of the ARIS-concept include variable modelling languages, e.g. EPC for illustrating the collaborative business processes. But there are extensions needed concerning the requirements of modelling collaborative enterprises like new role-concepts or the problem of depicting internal and external views of the same business process. ARIS has strong top-down process modelling and integration capabilities, but lacks expressiveness, view management and language constructs for other aspects, and does not support the "big picture" created by other approaches.

2. Enterprise Interoperability

Since the beginning of 2000s, the European Commission has proposed to identify the problematic/approach relating to the development of enterprise software applications. Many research projects have contributed to Enterprise Interoperability (EI) development that mainly concentrates on EI architectures, models, methodologies, and operational solutions. Based on the results of these research projects, numerous enterprise interoperability solutions have been tested and implemented to help enterprises to connect and to collaborate with their business partners in an extended and networking enterprise.

2.1 Definitions

In [Chen et al, 2002] and [Chen et al, 2004] authors had reviewed several definitions on interoperability. Interoperability is the ability of a system to understand another system and use its functionalities. The word ''inter-operate'' implies that one system performs an operation on behalf of (or for) another system. From software engineering point of view, interoperability means that two co-operating software systems can easily work together without a particular interfacing effort. It also means establishing communication and sharing information and services between software applications regardless of hardware platform(s). In other words, it describes whether or not two pieces of software from different vendors, developed with different tools, can work together. The definition of Interoperability in IEEE is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [IEEE, 1990]. According to [IDEAS, 2003], Enterprise interoperability is achieved if the interaction can, at least, take place at the three levels: data, application and business process. These definitions describe interoperability from various different aspects: interoperability's behavior, information interoperability, or software application interoperability. In addition, the definition from IDEAS focuses not only on information interoperability, but also on business processes interoperability.

From these definitions we can regard Enterprise Interoperability as the ability to communicate and exchange information, use exchanged information, and access functionalities of a third

system.

However, some researches considered that these definitions need to be extended to cover the additional interoperability issues in the enterprises. As a result, some new definitions of Enterprise Interoperability were given in different projects. Enterprise Interoperability Research Roadmap (EIRR) define Enterprise Interoperability as "a field of activity with the aim to improve the manner in which enterprises, by means of Information and Communications Technologies (ICT), interoperate with other enterprises, organizations, or with other business units of the same enterprise, in order to conduct their business. This enables enterprises to, for instance, build partnerships, deliver new products and services, and/or become more cost efficient" [Charalabidis et al., 2008].

European Interoperability Framework defines interoperability as "the ability of information and communication technology (ICT) systems and of the business processes they support to exchange data and to enable the sharing of information and knowledge" [IDABC, 2008]. It also indicates "Interoperability is the ability of disparate and diverse organizations to interact towards mutually beneficial and agreed common goals, involving the sharing of information and knowledge between the organizations via the business processes they support, by means of the exchange of data between their respective information and communication technology (ICT) systems" [IDABC, 2008].

These definitions involve interoperability between organizational units and business processes and units either within distributed enterprises or within an enterprise network. In a word, Enterprise Interoperability is perceived as a capacity of two or more enterprises, including all the systems within their boundaries and the external systems that they utilize or are affected by, in order to cooperate seamlessly, in an automated manner, in depth of time for a common objective [ENSEMBLE, 2011] [Gonçalves et al., 2012].

2.2 Dimensions

To better understand the Enterprise interoperability concept, to define and position our research theme, it is necessary to study various dimensions of enterprise interoperability. Those dimensions representing problems, issues and concerns of EI research and development are usually structured and represented in enterprise interoperability frameworks. Figure 16 shows the INTEROP Enterprise interoperability Framework (now CEN/ISO 11354 standard) [Chen et al., 2006] with its three main dimensions.

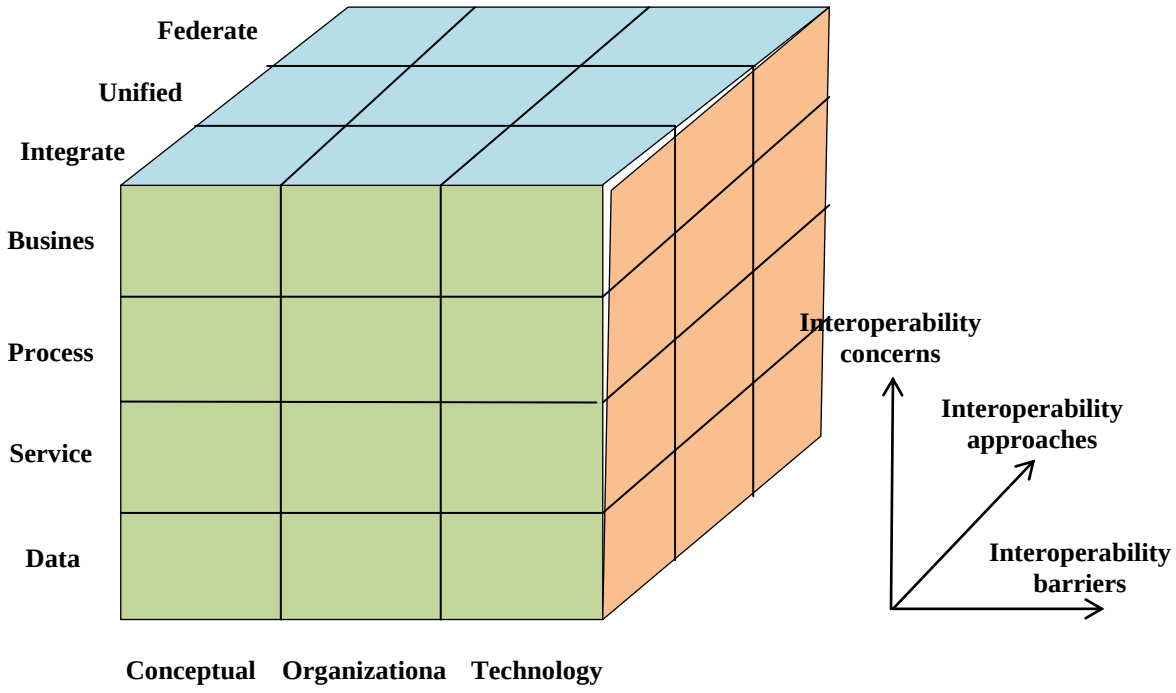


Figure 16 Enterprise Interoperability Framework

This framework consists of three basic dimensions:

- **Interoperability concerns** which defines the content of interoperation that may take place at various levels of the enterprise (data, service, process, business).
- **Interoperability barriers** which identifies various obstacles to interoperability in three categories (conceptual, technological, and organizational)
- **Interoperability approaches** which represent the different ways in which barriers can be removed (integrated, unified, and federated).

Interoperability concerns and interoperability barriers can constitute the interoperability problem space. The intersection of an interoperability barrier and an interoperability concern is the set of interoperability problems having the same barrier and concern. In order to constitute the solution for the interoperability problem, the interoperability approaches are imperative.

2.3 Approaches and Frameworks

2.3.1 **IDEAS interoperability framework**

The IDEAS interoperability framework (figure 17) was developed by IDEAS project. The framework was intended to reflect the view that ‘‘Interoperability is achieved on multiple levels: inter-enterprise coordination, business process integration, semantic application integration, syntactical application integration and physical integration’’. In the business layer, all issues related to enterprise’s organization and management are addressed. It includes the way an enterprise is organized, how it operates to produce value, and how it manages its relationships (internally with its personnel and externally with partners, customers and suppliers). Interoperability at this level should be seen as the organizational and operational ability of an enterprise to factually cooperate with other enterprises. The business layer includes the decisional model, the business model and business processes. The decisional model of an enterprise defines what/how decisions are taken and the degree of responsibility

of each operating unit, role and position. The business model is the description of the commercial relationships between an enterprise and the way it offers products or services to the market. Business processes are the set of activities that deliver value to one's customers [Athena, 2003]. The knowledge layer is concerned with acquiring, structuring and representing the collective/personal knowledge of an enterprise. It includes knowledge of internal aspects such as products, the way the administration operates and controls, how the personnel is managed and so on, but also of external aspects such as partners and suppliers, laws and regulations, legal obligations and relationships with public institutions. Interoperability at knowledge level should be seen as the compatibility of the skills, competencies and knowledge assets of an enterprise with those of other enterprises. This layer addresses the methods and tools that support the elicitation, gathering, organization and diffusion of business knowledge within an enterprise. The Knowledge layer includes several models. The organizational model can define the roles within – for example – the internal organization, the value chain, and a network of enterprises or a constellation. A skills-competency model defines the capability of an organization and of its employees to perform a certain job under certain working conditions. Enterprise's knowledge assets are the capital of the organization formalized in terms of procedures, norms, rules and references. The ICT systems layer is concerned with the ICT solutions that allow an enterprise to operate, make decisions, and exchange information within and outside its boundaries. The overall execution of the enterprise application will be orchestrated by the business process model identified in the top layer and formally (i.e. unambiguously) represented and stored in the middle (knowledge) layer. Interoperability at ICT systems level should be seen as the ability of an enterprise's ICT systems to cooperate with those of other external organizations. It is concerned with the usage of ICT to provide interoperation between enterprise resources (i.e. software, machines and humans). The interoperation has to be established by the supply of information through inter- and intra-system communication. The ICT layer includes various areas such as solution management, workplace interaction, application logic, process logic and data logic. Solution management is about the tools and procedures required to administer an enterprise system. This includes role and policy management monitoring and simulation tools. Workplace interaction refers to the interaction of the human user with the system, which could be described through input, output and navigation. Application logic describes the computation carried out by an enterprise system to achieve a business result. Process logic is the order (i.e. step-by-step) in which an application (or a subset) is carried out. Data logic describes what data is required and produced by an enterprise system during its lifecycle. This includes repository services and content management. The semantic dimension cuts across the business, knowledge and ICT layers. It is concerned with capturing and representing the actual meaning of concepts and thus promoting understanding. The holistic perspective on interoperability requires considering semantics on each layer of an enterprise. For enterprises that want to collaborate with each other and that need interoperability on a specific layer, it is

	Framework 1st Level	Framework 2nd Level	ONTOLOGY	QUALITY ATTRIBUTES					
			Semantics	Security	Scalability	Evolution			
ENTERPR. MODEL	Business	Decisional Model							
		Business Model							
		Business Processes							
	Knowledge	Organisation Roles							
		Skills Competencies					QUALITY ATTRIBUTES		
		Knowledge Assets					Performance	Availability	Portability
ARCHITECT PLATFORM	Application	Solution Management							
		Workplace Interaction							
		Application Logic							
		Process Logic							
	Data	Product Data							
		Process Data							
		Knowledge Data							
		Commerce Data							
	Communication								

Figure 17 IDEAS Interoperability Framework

of prime importance to create a mutual understanding [Athena, 2003]. To ensure that semantics are exchangeable and based on a common understanding, ontology and annotation formalism for meaning can be used. Quality attributes is a supplementary dimension of the framework. Business considerations determine qualities that must be accommodated in a system. These qualities are over and above that of functionality, which is the basic statement of the system's capabilities, services and behaviors. The considered attributes are: security, scalability, portability (both data and applications), performance, availability, and evolution. It must be underlined that the achievement of any quality attribute will have an effect, sometimes positive and sometimes negative, on the achievement of other quality attributes [IDEAS, 2002].

2.3.2 **LISI approach**

LISI (levels of information systems interoperability) approach is regarded as the first significant initiative of Enterprise Interoperability. It is developed by C4ISR Architecture Working Group (AWG) during 1997. The purpose of LISI is to provide the US Department of Defense (DoD) with a maturity model and a process for determining joint interoperability needs, assessing the ability of the information systems to meet those needs, and selecting pragmatic solutions and a transition path for achieving higher states of capability and interoperability [C4ISR, 1998]. A critical element of interoperability assurance is a clear prescription of the common suite of requisite capabilities that must be inherent to all information systems that desire to interoperate at a selected level of sophistication. Each

level's prescription of capabilities must cover all four enabling attributes of interoperability known as PAID, namely: procedures, applications, infrastructure (hardware, communications, security and system services) and data.

Nature of Operational Information Interaction	Corresponding Interoperability Level		Implications			
			P	A	I	D
Cross-Domain Interactive Manipulation	Enterprise	4	Enterprise Level	Interactive	Multiple Topologies	Enterprise Model
Shared Applications & Databases	Domain	3	Domain Level	Groupware	World Wide Networks	Domain Model
Complex Media Exchange	Functional	2	Program Level	Desktop Automation	Local Networks	Program Model
Simple Electronic Exchange	Connected	1	Local/Site Level	Standard System Drivers	Simple Connection	Local
Manual Gateway	Isolated	0	Access Control	N/A	Independent	Private

Figure 18 LISI reference model

The LISI reference model also provides the common vocabulary and structure needed to discuss interoperability between systems. At each level, a word or phrase highlights the most important aspect of PAID needed to achieve that level. For example, a system targeting interactions with other systems working at Level 3 (domain level in an integrated environment) must build toward the specific set of capabilities that underlie the PAID thresholds of the LISI reference model at level 3 (domain level procedures, groupware applications, access to world wide networks and domain data models). Although each attribute (PAID) is significant and must be considered in defining a level of interoperability, the significance and relative impact of the contributions from each attribute varies by level [C4ISR, 1998]. Besides this LISI reference model, a LISI interoperability maturity model and a practical assessment process for determining the interoperability maturity level of a given system or system pair is also defined. For more detail, see [C4ISR, 1998]. The LISI approach, although built with generic concepts and models, is focused on developing interoperability in US military sector. However, it is also used as a basis to elaborate other interoperability maturity models such as for example organizational maturity model [Clark and Jones, 1999] and enterprise interoperability maturity model [Athena, 2005].

2.3.3 ATHENA interoperability framework

The ATHENA Interoperability Framework (AIF) provides a compound framework and associated reference architecture for capturing the research elements and solutions to interoperability issues that address the problem from different perspectives of the enterprise.

2.3.3.1 Interoperability reference architecture

The ATHENA Interoperability Framework defines an interoperability reference architecture that relates the modelling solutions coming from the three different research areas of ATHENA, namely enterprise modelling, architectures and platforms, and ontology. The

following figure illustrates the reference architecture that focuses on the provided and required artifacts of two collaborating enterprises.

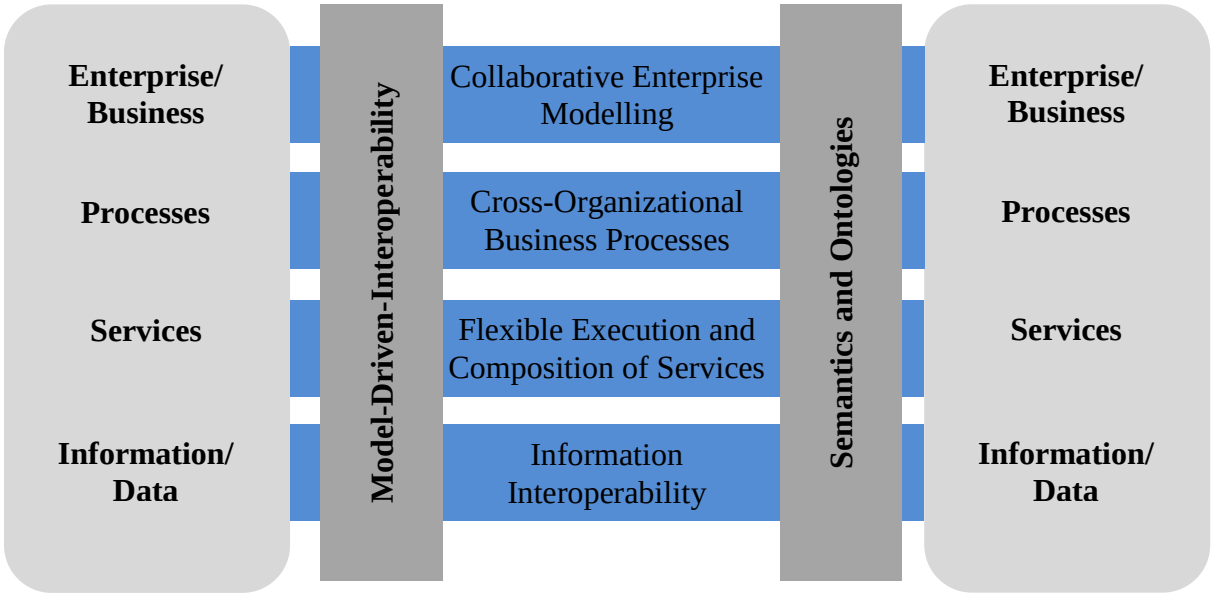


Figure 19 Athena Interoperability Reference Architecture

Interoperations can take place at the various levels:

- Interoperability at the enterprise/business level should be seen as the organizational and operational ability of an enterprise to factually co-operate with other, external organizations in spite of e.g. different working practices, legislations, cultures and commercial approaches.
- Interoperability of processes aims to make various processes work together. A process defines the sequence of the services (functions) according to some specific needs of a company.
- Interoperability of services is concerned with identifying, composing and executing various applications (designed and implemented independently). Services are an abstraction and an encapsulation of the functionality provided by an autonomous entity.
- Interoperability of information/data is related to the management, exchange and processing of different documents, messages and/or structures by different collaborating entities.

For each of these levels we prescribe a model-driven interoperability approach where models are used to formalize and exchange the relevant provided and required artefacts that must be aligned and made compatible through negotiations and agreements.

- Collaborative enterprise modelling concerns the exchange and alignment of knowledge models for describing the processes, organizations, products and systems in the collaboration context.
- Modelling of cross-organizational business processes focuses on defining process views that describes the interactions between two or more business entities.
- Flexible execution and composition of services is concerned with identifying,

composing and executing various applications.

- Information interoperability is related to management, exchange and processing of different documents, messages and other information structures.

To overcome the semantic barriers which emerge from different interpretations of syntactic descriptions, precise, computer processable meaning must be associated with the models expressed on the different levels. It has to be ensured that semantics are exchangeable and based on common understanding in order to enhance interoperability. This can be achieved using ontologies and an annotation formalism for defining meaning in the exchanged models. The model-driven interoperability and the semantics and ontologies approaches to interoperability cut across the four levels and focus on integration of the corresponding interoperability approaches at these levels.

2.3.3.2 Structure of the framework

The ATHENA Interoperability Framework (AIF) is structured into three main parts:

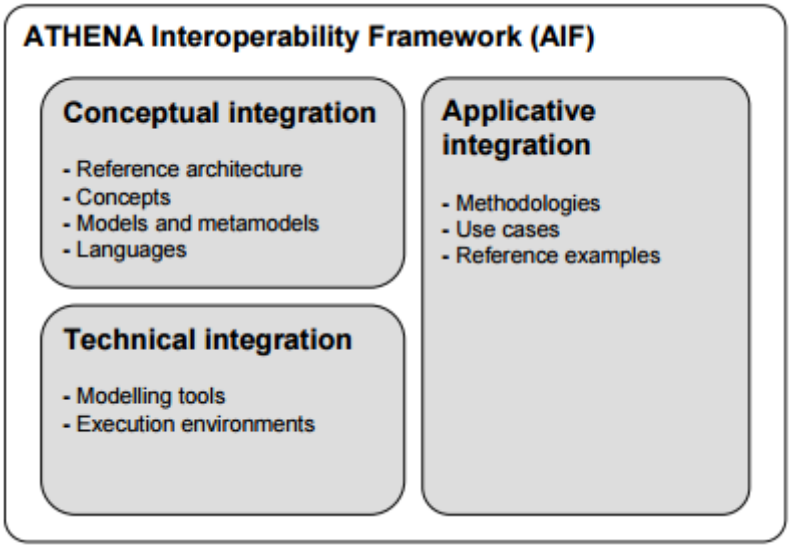


Figure 20 Structure of the AIF

- Conceptual integration which focuses on concepts, metamodels, languages and model relationships. The framework defines an interoperability reference architecture that provides us with a foundation for systemizing various aspects of interoperability.
- Applicative integration which focuses on methodologies, standards and domain models. The framework defines a methodology framework that provides us with guidelines, principles and patterns that can be used to solve interoperability issues.
- Technical integration which focuses on the software development and execution environments. The framework defines a technical architecture that provides development tools and execution platforms for integrating processes, services and information.

2.4 Conclusion on Enterprise Interoperability

In this section, several enterprise interoperability frameworks have been studied. These frameworks differ in their definition of interoperability depending on the angle (view) from which interoperability is examined. IDEAS defined Interoperability as the "ability of interaction between enterprise software applications". In the LISI approach, interoperability

was defined as “the ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces, and to use the services exchanged to enable them to operate effectively together”. Finally the Athena interoperability framework expressed interoperability as the following “interoperability occurs when there is the capacity of satisfactorily performing one or more operations not withstanding that control mechanisms, objects and/or tools do not belong to the same owner and/or technological and normative paradigm”.

3. Model Driven Development

Model Driven Development (MDD) is a software engineering paradigm where models are the core asset. They are used to specify, simulate, test, verify, and generate code for application to be built. Since models are the central artifacts in MDD, the quality of generated code and software is directly dependent on the quality of models. Ideas and business needs are collected at high abstraction levels and represented into models. These models are later transformed into code in lower abstraction levels.

3.1 MDA

3.1.1 Overview

Model Driven Architecture (MDA) has been defined and adopted by the Object Management Group (OMG) in 2001, and updated in 2003 [OMG, 2003]. It is designed to promote the use of models and their transformations to consider and implement different systems. The MDA has three major goals, which are portability, interoperability and reusability. The MDA starts with the well-known and long established idea of separating the specification of the operation of the system from the details of the way the system uses the capabilities of its software execution platform (e.g. J2EE, CORBA, Microsoft .NET and Web services). The MDA builds on six basic concepts -- System, Model, Architecture, Viewpoint, View and Platform. System means existing or planed system, which may include a program, a single computer system or some combination of parts of different systems. Model is a description or specification of the system modelled and its environment for some certain purpose. Architecture is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors. Viewpoint is a technique for abstraction using a selected set of architectural concepts and structuring rules. View is a representation of the system from the perspective of a chosen viewpoint. Platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.

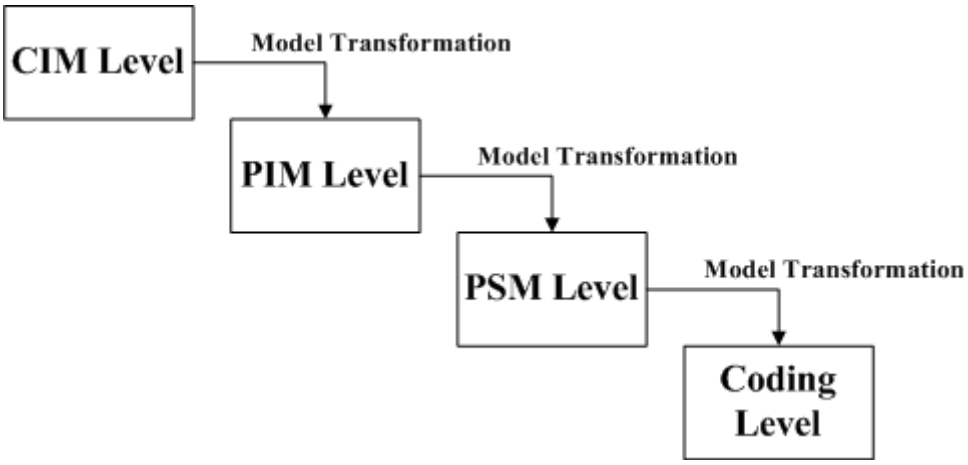


Figure 21 OMG's Model Driven Architecture

The MDA defines four levels according to different viewpoints, which go from general considerations (conceptual level) to specific ones (implementation level).

- CIM Level (Computation Independent Model) is a view of a system from the computation independent viewpoint. It focuses on the whole system and its environment. It is also named "domain model". It describes all work field models (functional, organizational, decisional, process, etc.) of the system with a vision independent from implementation.
- PIM Level (Platform Independent Model) is a view of a system from the platform independent viewpoint. It models the sub-set of the system that will be implemented, but does not show the details of its use of its platform. It might consist of enterprise, information and computational viewpoint specifications.
- PSM Level (Platform Specific Model) is a view of a system from the platform specific viewpoint. It takes into account the specificities related to the development platform. It combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.
- Coding Level (Implementation) is last level, consisting in coding enterprises applications (ESA: Enterprise Software Application). It is also a specification, which provides all the information needed to construct a system and to put it into operation.

As the name shows, "Model-driven" means using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification. Thus, the models of these four levels can be transferred to others under certain order and rules. Model transformation is the process of converting one model to another model of the same system. For example, model transformation from PIM to PSM, the input to the transformation is the marked PIM (a certain mapping assigned) and the mapping (specification for transformation under a particular platform). The result is the PSM and the record of transformation.

3.1.2 **MDA for Reuse and Interoperability**

As mentioned in the overview, MDA provides a systematic architecture to model a system, which can bring amount of advantages including reduction of development cost and complexity and increase of interoperability and reuse. As the enhancement of interoperability and reuse is the most promoted advantages of the MDA [OMG, 2003], and also major concern of this research, so this section will describe how MDA supports interoperability and

reuse. Concerning the MDA for reuse, most of the time, it takes place at these levels or between these levels. For example, reuse of the work field models from an existing CIM to other CIMs; reuse of entities and data types from a PIM to other PIMs; Use of UML profile entities and data types in many PIMs; Reuse of a given PIM as the model for many differing PSMs and implementations; reuse functional module in one PSM to other functional module within this PSM or to other PSMs; and etc. The examples show that the models being reused are general, flexible. They are only focus on one specific problem, and they remove the distraction and complexity. In a word, to reuse the model entities and types defined in an existing MDA model as the basement for other different business environments, technologies or platforms implementation can reduces development time and effort. Concerning MDA for interoperability, from intra-system MDA model point of view, the interoperability ability of MDA is not so obvious. However, from inter-system point of view, it will be very clear. As the MDA model transformation shows that, the model transformation starts from PIM to PSM, than to implementation depending on different techniques and platforms. Because PIM model is an abstract model contains enterprise, information and computational viewpoint specifications and includes the mappings to the implementation technology, if two system implementations are derived from the same PIM, then a bridge between these two implementations can be generated based on those known and standardized clues. In this way, the bridge enables the interoperability between these two system implementations. This example shows that to reuse the existing entities, types with a given PIM to guide a new implement across different technologies or platforms, a mapping or relationship among those implementations is concealed. Then, because the MDA around open, supported standards allows all models, data types and entities to be represented in a single, consistent manner, the interoperability of those implementations can be achieved.

Actually, to reuse or to map the model in PIM model showed in the example is just one way to achieve the interoperability. The interoperability can be achieved in even more abstract level, such as remove the business duplicate issues in CIM level, or in more detail level, such as adjust the function module in PSM level. The agile MDA model allows developer to realize the interoperability in different levels. This must be the original idea of Model Driven Interoperability, which will be introduced in next section.

3.2 MDI

As previous section mentioned, the MDA provides a way for developing modern enterprise applications and software systems, meanwhile, it also provides a better way of addressing and solving interoperability issues compared to earlier non-modelling approaches. In addition, from an interoperability point of view, most of the enterprises build their information system by using MDA, so it seems that MDA is a good solution for overcoming the interoperability barriers [Ullberg et al., 2007]. As a result, the researchers believe that an interoperability framework based on MDA can provide guidance on how model driven development (MDD) should be applied to address interoperability. Thus, Model Driven Interoperability (MDI) framework is created for how to apply Model Driven Development (MDD) in software engineering disciplines in order to support the business interoperability needs of an enterprise [Elvesæter et al., 2007]. It is a model driven method that considers interoperability problems at the enterprise model level instead of only at the coding level. It provides a foundation, consisting of a set of reference models. Figure 22 shows the reference model of MDI approach which performs different abstraction in each MDA levels. Between each level of models, the successive model transformations are carried out to reduce the gap existing between enterprise models and code level. The models at the various levels may be semantically annotated (such as reference ontology) which helps to achieve mutual

understanding on all levels. The mutual understanding also helps to achieve model interoperability horizontally between different enterprises' model in homologous level.

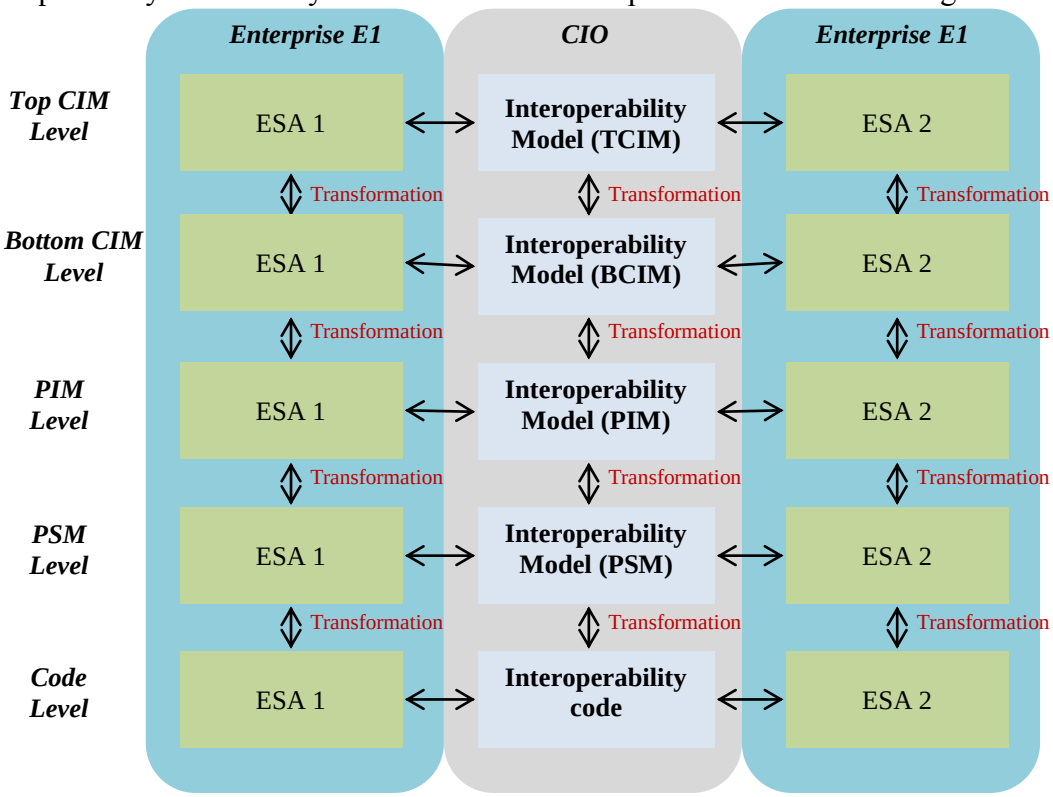


Figure 22 Reference model for MDI

The concepts of this method were realized in the Task Group 2 (TG2) of INTEROP-NoE project by defining an approach inspired by the OMG MDA concepts [Bourey et al., 2007]. The goal of MDI is to tackle the interoperability problems at each abstraction level defined in MDA and to use model transformation technique to link both vertically the different levels of the MDA abstraction and horizontally the corresponding models of the systems to interoperate. The main goal of MDI, based on model transformation, is to allow a complete follow-up from the expression of requirements to the coding of solutions and also to provide a greater flexibility thanks to the automation of these transformations.

In the context of TG2, experimentations have been realized and in particular the feasibility study to transform GRAI Methodology [Chen et al., 1997] [Doumeingts et al., 2001] Models to UML models between CIM and PIM levels [Bourey et al., 2007]. These works are complemented by additional works realized in the context of ATHENA [ATHENA, 2003] to define UML profiles to take into account also the Service Oriented Architectures (SOA) at the PIM level [Gorka et al., 2007].

3.3 Conclusion on Model Driven Development

Model Driven approach is essential to allow the implementation of services in coherence with its definition at the business level using enterprise models. MDA defines the modeling levels and specifies the goals to reach at each level but without mentioning how to model or which modeling language to be used. In addition, interoperability barriers represent a key issue for the development of collaborative networks and for the exchange of data between networked organizations but they are only tackled at all abstraction level by MDI. Therefore, it is

necessary to develop a dedicated model driven approach defining accurately each modeling level with proposing modeling languages, interoperability and the transformation mechanisms from one level to another.

4. Modelling Languages

4.1 GRAI Extended Actigram

GRAI Extended Actigram is a process modeling language developed in the frame of GRAI methodology. It offers many constructs to model different Enterprise functions and operations. This formalism, which is an extension of IDEF0 [NIST, 1993], makes it possible to model Enterprise functions with a high semantic level. GRAI Extended Actigrams are composed of Activities and Control Flow, Resource Flow, Input Flow, and Output Flow, like IDEF0, but they also provide logical operators in order to synchronize flows between activities. In GRAI Extended Actigrams, flows can be characterized as Product or Information flows. Two kinds of resources are taken into account: Human or Material. Figure 23 shows an example of a 'Painting Check Process' in order to present the different types of constructs that it is possible to use when creating GRAI Extended Actigrams.

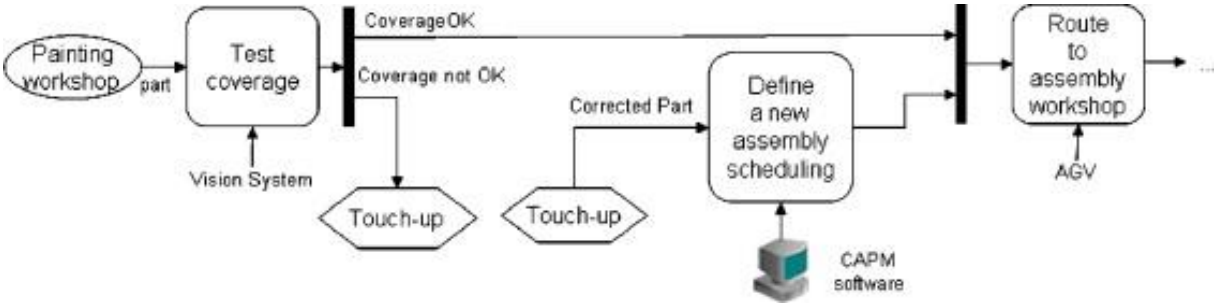


Figure 23 GRAI Extended Actigram for 'Painting Check Process'

This formalism doesn't provide any kind of formal specification necessary for farther development or implementation activities.

4.2 BPMN

The Business Process Modeling Notation (BPMN) [OMG-2 2011] is a standard defined by the Object Management Group (OMG) for modeling business processes. The development of BPMN was influenced by the demand for a graphical notation that complements the BPEL [Andrews T. et al] standard for executable business processes BPMN was first developed by the Business Process Management Institute (BPMNI), now merged with OMG, and released to the public in May, 2004. BPMN was adopted as an OMG standard in February, 2006. The primary goal of the BPMN effort was 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. BPMN targets both business analysts and software architects to collaboratively design, deploy and monitor business processes. It enables analysts to freely design the processes and developers to add necessary technical details afterwards. Due to its maintenance by the Object Management Group (OMG) and its adoption as an ISO standard (ISO 19510:2013), BPMN also meets the requirement to use a generally accepted notation, which guarantees certain sustainability. Although BPMN offers a wide range of modelling elements, it also defines a basic set of core elements, which simplifies the modelling and understanding of complex

business processes

BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance. A BPD is made up of a set of graphical elements. The elements were chosen to be distinguishable from each other and to utilize shapes that are familiar to most modelers. For example, activities are rectangles and decisions are diamonds. Graphical aspects of the notation are organized into specific categories. This provides a small set of notation categories so that the reader of a BPD can recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram. The five basic categories of elements are: Flow Objects, Data, Connecting Objects, Swimlanes, and Artifacts. Flow objects are the main graphical elements to define the behavior of a business process. It consists of activities, gateways, and events. Data is represented with the four elements Data Objects, Data Inputs, Data Outputs and Data Stores. Connecting objects as the name implies are used to connect the activities and other elements with each other using different arrows which represent messages and associations between them. This core set of elements define the control flow perspective of processes. There are four connecting Objects: Sequence Flows, Message Flows, Associations, and Data Associations. Different modeling elements are grouped through Swimlanes which are pools and lanes. A Pool is used to represent process participants while lanes are used to partition these participants and their activities from one to another. A process participant can either be organizational entities within an organization or different organizations for collaboration in a process. Mostly, organizational perspective is provided by using Swimlanes constructs. Artifacts are used to provide additional information about the Process. There are two standardized Artifacts, but modelers or modeling tools are free to add as many Artifacts as necessary. There could be additional BPMN efforts to standardize a larger set of Artifacts for general use or for vertical markets. The current set of Artifacts includes: Group and Text Annotations.

Modern business process modelling languages like BPMN offers more constructs to represent real-world situations than their predecessors, e.g. IDEF [NIST 1993] or Petri nets [Narahari, 1999]. BPMN offers 50 modelling constructs, ranging from Task and Sequence Flow to Compensation Associations and Transaction Boundaries [zur Muehlen et al]. However, the apparent increase in expressiveness is accompanied by an increase in language complexity. The apparent complexity of the BPMN standard seems to be similar to the UML standard, which raises a number of questions: Are BPMN users able – and willing – to cope with the complexity of the language? Does the separation into core and extended constructs provided by the specification hold in modelling practice? And – really – how exactly is BPMN used in practice? Authors in [zur Muehlen et al] tried to answer these questions by analysing BPMN diagrams collected from different data sources. Authors observed that the distribution of BPMN constructs shows that BPMN – as many natural languages – has a few essential constructs, a wide range of constructs commonly used, and an abundance of constructs virtually unused. Based on this observation, they concluded that training and usage guidelines can be designed to reduce the complexity of the language to inexperienced analysts and to deliberately build such models that can safely be assumed to depict the core essence of a process without adding too much complexity.

4.3 DEVS Formalism

Since the early 1970s, the modeling and simulation (M&S) Community has tried to define different formalisms for varied systems specifications. The DEVS formalism was defined to bring coherence and to unify the field of discrete-event M&S with formal rigor and an underlying system's theoretical framework. DEVS stands for Discrete EVent System specification, a formalism introduced first by Bernard Zeigler [Zeigler 1976]. A DEVS model processes an input event trajectory and –according to that trajectory and its own initial conditions– it provokes an output event trajectory.

4.3.1 **Atomic DEVS**

An *atomic* DEVS model is defined by the following structure:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

where:

- X is the set of input event values
- Y is the set of output event values
- S is the set of state values
- δ_{int} : S → S is the internal transition function
- $\delta_{ext} : Q \times X \rightarrow S$ is the external transition function

  $Q = \{(s, e) : s \in S, e \in [0, ta(s)]\}$ is the total state set, and e is the elapsed time since the last transition
- λ: $S \rightarrow Y$ is the output function
- ta: $S \rightarrow R_0^+ \cup \infty$ are functions which define the system dynamics.

For a discrete event model described by an atomic DEVS M, the behavior is uniquely determined by the initial total state $(s_0, e_0) \in Q$ and is obtained by means of the following iterative simulation procedure. Each possible state s ($s \in S$) has an associated *Time Advance* computed by the *Time Advance Function* ta(s): $S \rightarrow R_0^+$. The *Time Advance* is a non-negative real number saying how long the system remains in a given state in absence of input events. Thus, if the state adopts the value s_1 at time t_1, after $ta(s_1)$ units of time (i.e. at time $ta(s_1) + t_1$) the system performs an *internal transition* going to a new state s_2. The new state is calculated as $s_2 = \delta_{int}(s_1)$. Function δ_{int} (δ_{int} : S → S) is called *Internal Transition Function*. When the state goes from s_1 to s_2 an output event is produced with value $y_1 = \lambda(s_1)$. Function λ (λ: S → Y) is called *Output Function*. In that way, the functions ta, δ_{int} and λ define the autonomous behavior of a DEVS model. When an input event arrives the state changes instantaneously. The new state value depends not only on the input event value but also on the previous state value and the elapsed time since the last transition. If the system arrived to the state s_2 at time t_2 and then an input event arrives at time t_2+e with value x_1, the new state is calculated as $s_3 = \delta_{ext}(s_2, e, x_1)$ (note that $ta(s_2) > e$). In this case, we say that the system performs an *external transition*. Function δ_{ext} ($\delta_{ext} : Q \times X \rightarrow S$) is called *External Transition Function*. No output event is produced during an external transition. After an external transition, the model is rescheduled and the process starts again, setting the elapsed time e to 0.

The behavior of an Atomic DEVS model is identified by the following:

- The time advance function (ta) which controls the timing of internal transitions, and usually this function just return the value of *sigma*
- The internal transition function which specifies the next state of the system after the time (*sigma*) given by ta has elapsed.
- The external transition function which specifies how the system changes state when an input is received. It places the system in new state and consequently a new *sigma* thus scheduling it for a next internal transition
- The output function which generates an external output just before an internal transition takes place.

In summary, *sigma* holds the time remaining to the next internal transition. This is precisely the time advance value to be produced by ta. In the absence of external events, the system stays in the current state for the time given by sigma. The time advance function can take any real number between 0 and ∞. A state for which ta(s) = 0 is called transient state. While if ta(s) = ∞, s is said to be a passive state.

4.3.2 **Coupled DEVS**

A coupled DEVS N is specified by a 7-tuple:

$$N = (X, Y, D, \{M_i\}, \{I_j\}, \{Z_{j,k}\}, \gamma)$$

Where

- X is the input set
- Y is the output set
- D is the set of component indexes
- $\{M_i | i \in D\}$ is the set of components, each M_i being an atomic DEVS
- $\{I_j | j \in D \cup \{self\}\}$ is the set of all influencer sets
- $\{Z_{j,k} | j \in D \cup \{self\}, k \in I_j\}$ is the set of output to input translation functions, where

  $Z_{j,k}: X \to X_k, if\ j = self$

  $Z_{j,k}: Y_j \to Y, if\ k = self$

  $Z_{j,k}: Y_j \to X_k, otherwise$
- $\gamma: 2^D \to D$ is the select function

Sets X and Y are produced sets which formalize multiple I/O ports. Each atomic DEVS in the network is assigned a unique identifier in the set D. This corresponds to model names or references in a modelling language. The coupled-DEVS N itself is referred to by means of $self \notin D$. This provides a natural way of indexing the components in the set $\{M_i\}$ and to describe the sets $\{I_j\}$, which isexplicitly describes the network structure, and $\{Z_{j,k}\}$.

4.4 Conclusion on Modeling Languages

GRAI Extended Actigram and BPMN are two business process modeling languages with a difference in the information and the level of details represented in their models. GRAI Extended Actigram doesn't possess a public formal specification or a standard metamodel that can be used for development purposes. In addition BPMN doesn't propose the

representation of specific performance indicators such as time and costs. On the other hand, DEVS formalism is a modeling and simulation (M&S) formalism, with well formulated specification and designated to capture and simulate the behavior of systems on time and cost bases.

5. Simulation Tools

5.1 Business Process Simulation Tools

Business processes are in a continuous improvement cycle in which design and redesign play an important role. Various possibilities to change a process are present and the best alternative design should replace the current process. Making an intuitive choice may lead to unpleasant surprises and lower process performance instead of yielding the expected gains. Simulation is one of the techniques suitable for the support of redesign. The simulation of business processes helps in understanding, analyzing, and designing processes. With the use of simulation the (re)designed processes can be evaluated and compared. Simulation provides quantitative estimates of the impact that a process design is likely to have on process performance and a quantitatively supported choice for the best design can be made

5.1.1 **ARIS Simulation**

ARIS Simulation is a professional tool for the dynamic analysis of business processes. It is an integral part of the ARIS Toolset; processes recorded in the ARIS Toolset are used as the data basis for business process simulation. ARIS Toolset is developed by IDS Scheer AG (see www.ids-scheer.nl) and can be classified as an enterprise modelling tool with a strong emphasis on business processes. Enterprise modelling is supported by a number of different views (process, function, data, organization and product) and the modelling approach called ARIS House. The process modelling part supports the definition of business processes represented in Event-driven Process Chains (EPCs). Other modelling techniques supported in the ARIS House are, e.g. value chains (also to model the control flow), organization charts (to model relationships between resources), EPCs and function allocation diagrams (for supplementary information such as data and systems). The simulation functionality shows whether the specified processes are executable at all and it answers questions about throughput times and utilization levels of the resources, etc. When starting a simulation, the simulation module of the tool is started and the model is transferred. The simulation toolbar shows buttons for start and stop, one time step and simulation steps and options for animations. The simulation results are available in Excel spreadsheets and include statistics on events, functions, resources, processes and costs. Only raw data is available.

5.1.2 **Protos**

Protos is a modelling and analysis tool developed by Pallas Athena and it is mainly applied for the specification of in-house business processes. Protos is suitable to model well-defined Petri Net structures. Nevertheless, it also permits free hand specifications of business processes without formal semantics, e.g. to support initial and conceptual modelling. When formal Petri Net semantics have been applied, translation to various other process-based systems is feasible as well, e.g. to the workflow management system COSA and the workflow analyzer Woflan.

The main use of Protos is to define models of business processes as a step towards either the implementation of quality management systems, the redesign of a business process, communication enhancement between process stake holders or the implementation of workflow management systems. The process can be analyzed with respect to data, user and

control logic perspective, and by making use of simulation. The simulation engine is implemented in Protos version 7.0. The existing engine of the Petri Net based tool ExSpect has been integrated in the Protos environment and it facilitates the simulation of the business process as has been specified in the Protos model before. In addition to the standard process specification, simulation data can be added for tasks, connections and resources such as the (stochastic) processing time and the number of resources required. Furthermore, process characteristics are added such as the arrival pattern for cases and the number and length of simulation runs. The simulation result can be obtained from an Excel spreadsheet and includes mean and 90% and 99% confidence interval of utilization rates, waiting times, service times, throughput times and costs.

5.1.3 **Arena**

Arena is a general purpose simulation tool developed by Rockwell Automation. The Arena product family consists of a Basic Edition for uncomplicated processes and a Professional Edition for more complex large scale projects in manufacturing, distribution, processes, logistics, etc. The Professional Edition also provides (and allows definition of) templates for complex repetitive logic, e.g., for packaging and contact centers.

When opening the tool, a number of process panels are available, e.g., for basic and advanced processes and for reporting. The model can be created by drag and drop from the process panel to the model window. By double-clicking on the icons, options for the different building blocks can be set such as delay types, time units and the possibility to report statistics. Many more building blocks are available and can be attached when necessary.

When a model has been created and is completely specified (from the Arena viewpoint) and it is syntactically correct, it can be simulated. Warm-up and cool down periods can be specified, as well as run length and confidence intervals.

Several statistics are provided by default, but the larger part needs to be added manually by adding record building blocks where necessary. In a previous study, [de Vreede et al] considered the suitability of Arena to simulate business processes. They stated that a weak point in simulating business processes is the time consuming and complicated process to create simulation models. They took advantage of the possibility to develop their own template with predefined building blocks, which they considered to be successful in several simulation studies they carried out.

5.1.4 **Jbpm**

JBoss Jbpm is a very flexible business process engine which is available under the open source LGPL license2. The core of Jbpm is a light-weight, extensible workflow engine written in pure Java that allows you to execute business processes using the latest BPMN 2.0 specification. It can run in any Java environment, embedded in your application or as a service. On top of the core engine, a lot of features and tools are offered to support business processes throughout their entire life cycle.

BPM makes the bridge between business analysts, developers and end users, by offering process management features and tools in a way that both business users and developers like it. Domain-specific nodes can be plugged into the palette, making the processes more easily understood by business users. Jbpm supports adaptive and dynamic processes that require flexibility to model complex, real-life situations that cannot easily be described using a rigid process. We bring control back to the end users by allowing them to control which parts of the process should be executed, to dynamically deviate from the process, etc. jBPM is also not

just an isolated process engine. Complex business logic can be modeled as a combination of business processes with business rules and complex event processing. Jbpm can be combined with the Drools project to support one unified environment that integrates these paradigms where you model your business logic as a combination of processes, rules and events. Jbpm5 is the latest community version of the Jbpm project. It is based on the BPMN 2.0 specification and supports the entire life cycle of the business process (from authoring through execution to monitoring and management). The current Jbpm5 snapshot offers open-source business process execution and management, including:

- Embeddable, lightweight Java process engine, supporting native BPMN 2.0 execution
- BPMN 2.0 process modelling in Eclipse (developers) and the web (business users)
- Process collaboration, monitoring and management through the Guvnor repository and the web console
- Human interaction using an independent WS-HT task service
- Tight, powerful integration with business rules and event processing

5.1.5 **Bonita Open Solution**

BonitaSoft is a leading BPMS solutions (workflow) proposed in open source mode, located in France, China and the USA and is represented in more than 20 countries through its network of partners. BonitaSoft is the publisher of Bonita Open Solution, a platform BPMS (business process modelling simulation). Bonita Open Solution BPM suite is the most world downloaded open with more than 500,000 downloads in early 2011.

It combines three solutions in one: an innovative Studio for process modelling, a BPM and workflow engine, and a user interface. Standard simulation capability available in Bonita Open Solution allows loading parameters and provides execution simulation reports.

5.1.6 **Evaluation**

The evaluation of the previous simulation tools is based on four basic aspects: BPMN2.0 compatibility, Simulation capabilities (possibility to simulate time and cost aspects, use of different simulation scenarios, and animation or replay of the simulation), Result analysis capabilities (statistical results and easy to read formats), and product's license type (open source). The following table summarizes an evaluation of the tools with respect to the defined criteria.

Table 1 Evaluation of Business Process Simulation Tools

Criteria \ Tool	ARIS	Protos	Arena	Jbpm	Bonita
BPMN2.0 Compatible					
time/cost simulation					
Simulation scenarios					
Simulation animation					
Statistical results					
Easy to read formats					
Open source					

Not supported
Partially supported
Supported

5.2 DEVS Simulation Tools

Electing a target DEVS tool for model transformation requires a literature review of current DEVS Simulation tools. The DEVS group standardization maintains on its website the updated list of most used DEVS tools known by the DEVS community [Wainer 2013]. In [Hamri and Zacharewicz 2012], the authors have given a brief description and comparison of popular tools.

ADEVS was the first DEVS tool developed in C++ by the Arizona University. It consists in an ad-hoc simulator. DEVS abstract classes should be extended by users to define atomic and coupled models, and then the simulation can be launched. The drawback resides in the fact that users need programming skills to code the models.

DEVSJAVA is a Java framework in which the kernel simulator is ADEVS. It supports also modelling and simulation of DEVS with variable structures. However, at atomic level, the user should implement the corresponding DEVS behavior in Java (in our opinion the user has not enough skills to program his atomic models).

CD++ Builder is a DEVS modelling and simulation environment that integrates interesting features and facilities for the user. It allows modelling and simulation of other DEVS formalisms (cell-DEVS, Quantized-DEVS, etc.). It provides a DEVS graphical editor to model coupled and atomic models, and to encapsulate them through components for further reuse. Other DEVS tools are dedicated to specific areas. VLE, this is a C++ M&S framework that integrates heterogeneous models from different scientific fields. This integration is based on the agent paradigm. In addition, JDEVS is the Java implementation of a DEVS formal framework. It supports multi-modelling paradigms based on DEVS. It ensures the interoperability among the reused components. Also SIMSTUDIO can be considered, it is focused on a simplified DEVS editor for DEVS non Expert. The authors also investigate LSIS_DME that is focused on a graphical interface and code source generation in order to complete the model by complex Java functions.

5.3 Conclusion on Simulation Tools

The evaluation of business process simulation tools presented in this section is based on the

criteria we need in our work for the development or integration of a simulation tool. ARIS, PROTOS, and ARENA are discarded due to the lack of two basic criteria: BPMN2.0 compatible and open source. In the other hand, BONITA (open source) doesn't provide any animation support and it is not possible to integrate with other tools (eclipse based tools). A Jbpm eclipse plugin permits the integration of a Jbpm simulator with an eclipse based tool but this plugin doesn't support any animation feature. In addition, the examination of DEVS Simulation tools has faced functional and technical problems because of the absence of any support or user and technical manuals. These tools were difficult to use and were not adapted to our needs.

Model Driven Service Engineering Architecture (MDSEA), Extended Actigram Star (EA*), and Model Transformation

The previous chapters (chapter 1&2) have introduced the context of this thesis, existing problems we are targeting, and basic concepts in the domain of our research work. This chapter will start providing answers in the domain of servitization and Service System's modeling. ***M****odel* ***D****riven* ***S****ervice* ***E****ngineering* ***A****rchitecture is presented and its principles, concepts, and modeling languages for Service System's modeling are detailed.*

1. Service Systems' Modeling and Model Driven Approach

A Service System has the same structure like that of a Product System in a manufacturing enterprise, but it is oriented towards the realization of a service rather than product. How to model service systems and based on what concepts and methods? Service System modeling can be inspired from Enterprise Modelling concepts, models, methods and tools.

The advantage of Enterprise Modelling approach is the ability to precisely identify models' elements (concepts or constructs) using reference models and then to represent and 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, and IT. Enterprise modelling's techniques allows:

- 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 strong points and points to improve, using specific rules and taking into account the enterprise's strategy 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 Enterprise Modelling and by extension in Service System Modeling. One of the important lessons learned from applying Enterprise modelling in industry is the necessity to adapt two views: a global view which allows the capturing of global structure and understanding objectives and a local view which allows the modeling of detailed elements in a coherent way with the global view. Herbert Simon [March et Simon, 1963], one of the founder of System Theory, has claimed that "you can never know an enterprise or an organization, not only if you are not able to understand it as a whole but also if you are not able to represent its details and establish a link between its two views" (global and local views). Based on the understanding of System theory, we can comprehend the behavior of service systems and how to design, understand and represent Service Systems. The representation of these systems is based on models, which implies the necessity of modeling languages adapted to the nature of concepts and point of views to represent. We propose in the following section the Model Drive Service Engineering Architecture (MDSEA) inspired from the various enterprise modeling approaches presented in chapter 2 and based on field experience accumulated while implementing Enterprise modeling in enterprises.

2. MDSEA

The Model Driven Service Engineering Architecture (MDSEA) is inspired from MDA/MDI.

This methodology is proposed in the frame of the MSEE project [MSEE, 2011] that defines its first Grand Challenge as making SSME (Service Science, Management and Engineering) evolving towards Manufacturing Systems and Factories of the Future. MDSEA provides an integrated methodology dealing with modelling languages at various levels of abstraction to support service models and Service System's design and implementation. A relationship between MDSEA modelling levels and the Service System lifecycle phases (user-requirements, design and implementation) is established. One of the important innovations in MDSEA is to define the integration between domain components (IT, Organization/Human and Physical Means) at the Business Service Model (BSM) level in order to ensure that these integration aspects will be spread out at the other levels. In this sense, this is therefore considered as an adaptation and an extension of MDA/MDI approaches to the engineering context of product related services in virtual enterprise environment.

On the basis of MDA/MDI, the proposed MDSEA defines a framework for service system modelling around three abstraction levels: Business Service Model (BSM), Technology Independent Model (TIM) and Technology Specific Model (TSM) as presented in figure 24

Vertical decomposition: towards alignment from business to operational

Figure 24 shows that the interest of such architecture is on one hand to design and implement a service product and on the other hand to produce the dedicated service system coherent with business service models, represented using enterprise models. By examining TIM and TSM levels, we can observe how the methodology is differentiating between three kinds of resources categorized into IT, Human and Physical Means. The reason of such categorization is to tackle the different requirements of resources at the implementation stage of the service system. The implementation of resources detailed in TSM models allow the implementation of service systems and related service product through a cloud of services, i.e. a system in which the service provider (an enterprise inside the network) is not always recognized by the customer who is only focused on the service. The service maintenance, and decommission activities can also be ensured by different companies in the network without a real recognition by the customer. However, the dedicated virtual organization has also the property rights on the provided services

It is important to mention that the service system represented at each level of MDSEA is the same system but more or less detailed and taking into account more or less implementation constraints.

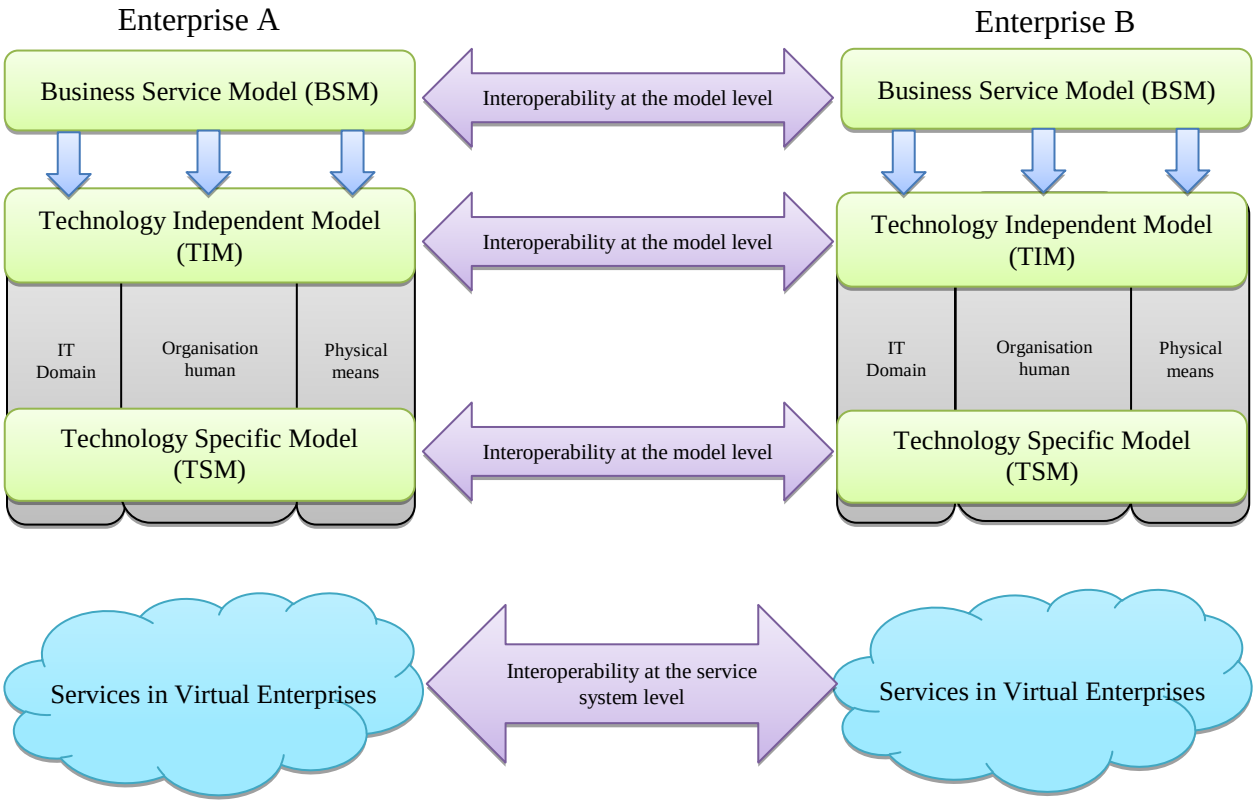


Figure 24 The MDSEA architecture applied in a service network of two enterprises

Horizontal alignment: towards an interoperability to ensure efficient collaboration between service network

Figure 24 shows the Collaboration between two enterprises collaborating together in order to produce a service. Collaboration between different entities can happen at different MDSEA abstraction levels (BSM, TIM, and TSM). The BSM models allow to represent the TO BE models of both entities and to align the interoperability of practices in terms of business processes models and decisions models. In MDSEA, interoperability is a Key factor for enterprises' collaboration. Enterprise models ensure not only interoperability of practices, but also between human resources and IT systems supporting these practices.

Figure 25 presents the three abstraction levels proposed by MDSEA and their correspondence in MDA.

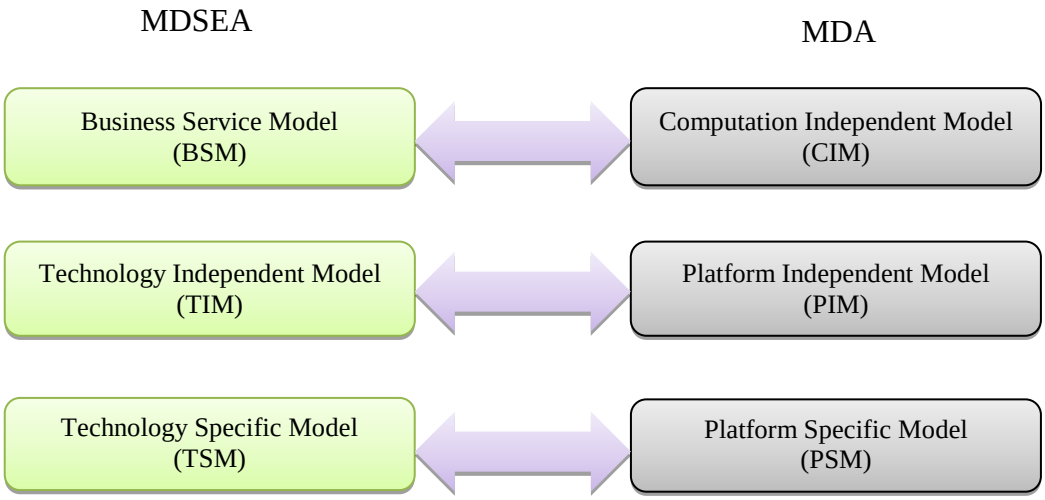


Figure 25 MDSEA vs MDA

2.1 Business Service Model (BSM)

BSM 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 and must reflect the business perspective of the service system. 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 who will build the service system. The BSM level allows also defining the link between the production of products and the production of services.

2.2 Technology Independent Model (TIM)

TIM delivers 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 which do not include technological details. More concretely, it focuses on the operational details while hiding specific details of any particular technology in order to stay independent from any technology, used for the implementation. At TIM level, the detailed specification of a service system's components are elaborated with respect to IT, Organization/Human and Physical means involved within the production of the service. This is important to mention that in comparison to MDA or MDI or SOMA (Service Oriented Modelling and Architecture), the objective of MDSEA is not only IT oriented and then this requires enabling the representation of human and technical resources from the BSM level. At TIM level, the representations must add some information in comparison to BSM models.

2.3 Technology Specific Model (TSM)

TSM enhances the specifications of the TIM model with details that specify how the implementation of the system uses a particular type of technology (such as, for example IT applications, Machine technology or a specific person). At TSM level, the models must provide sufficient details to allow developing or buying suitable software applications, hardware 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 applications, a TSM model enhance a TIM model with 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.). Based on the technical 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.

2.4 Proposed Modelling Languages

Based on the modelling levels just previously described, the methodology MDSEA proposed to associate relevant modelling languages at each level in order to represent confidently the existing system and the future service product and service system. To achieve this goal, the standards for process modelling are gaining more and more importance, which gave rise to several process modelling languages and tools to enhance the representation of enterprise processes. The level of abstraction required is important to choose the suitable modelling language.

It is obvious to say that the first specification step of a service to be established between two partners is crucial. At the BSM level, the modelling language must be simple to use, powerful and understandable by business oriented users. Moreover, this (or these) language(s) must cover process and decision with coherent models. The choice is affected by the capacity of the language to propose a hierarchical decomposition (global view to detailed ones); this is especially required at this level. Indeed, business decision-makers often have a global view of the running system and need languages allowing this global representation with few high level activities (process or decisions). This global view must be completed by more detailed activities models elaborated by enterprise sector responsible. These models are connected to top level models in a hierarchical and inclusive way. These are the principles of systemic and system theory which must be taken into account in the choice of the languages.

But it is also obvious that the choice of modelling languages is also subjective, depending on the experience of the languages' practitioners and on the wide dissemination of these languages within enterprises.

As for process modelling at business level, several languages exist. Extended Actigrams Star (EA*), extended from GRAI extended Actigram [Grangel 2008], that was itself derived from IDEF0 [NIST 1993], was chosen to model processes at BSM level due to its independence regarding IT consideration, its hierarchical decomposition and the fact it can model three supported resources: material, human and IT. It has been developed as an answer to previous issues encountered with GRAI extended actigram language regarding its interoperability. It intends to capture business process models at a high semantic level, independently from any technological or detailed specifications. Service Oriented Modelling and Architecture principles [Bell M. 2008] developed by IBM were also considered, but these languages are more IT oriented and thus were far away from our requirements. Moreover, GRAI Grid [Doumeingts G. 1998] was selected for modelling governance in a service system. GRAI Grid aims at proposing a cartography of company's decisions which controls business processes, as proposed for instance in the ISO 9000-2008 standard. The interest of GRAI Grid is to represent all decisions and their coordination, from the strategic to the operational levels. This representation is very important for business users because the results of decision making are also at the origin of performance evolution and achievement.

At the TIM level, BPMN 2.0 [OMG-2 2011] was chosen in particular because this language

offers a large set of detailed modelling construct, including IT aspects and benefits from the interoperability of many BPM IT platforms allowing the deployment and automated transformation to execution of BPMN processes. Moreover, BPMN enables also to represent human and technical resources which are required in the MDSEA principles of representation. BPMN has also the advantage to provide a meta-model developed by OMG which facilitates the implementation of the language. GRAI nets are proposed in order to detail the decision processes in coherence with the decisions identified in the GRAI Grid but with adding technical and organization information as the decision rules, the decision makers, and the decision support modules.

3. Extended Actigram Star (EA*)

Section 3.2 has explained and detailed MDSEA as a methodology for modeling service systems and assisting the shift of manufacturers towards servitization. In addition, as mentioned earlier MDSEA specifies the modeling languages to be used at every abstraction level (BSM, TIM, and TSM). The modeling of business processes at BSM is to be managed using the Extended Actigram Star (EA*) we developed for this purpose. This section introduced the EA* modeling language developed during this thesis for the purpose of modeling business processes.

3.1 Scope

The primary goal of Extended Actigram Star is to provide a common and explicit graphical notation for business process modelling. Such language is targeted to business oriented people, who need to describe and communicate high level business processes involving enterprises resources with the help of a simple and explicit formalism. In comparison to other initiatives such as BPMN2.0, Extended Actigram relies on a reduce set of graphical objects and focus on the "business" aspects of enterprise processes. By its simple and accessible syntax, Extended Actigram Star intends to reduce the gap between the ideation and the design of business process.

Extended Actigram Star has been developed in the frame of the European Integrated Project MSEE (Manufacturing Service Ecosystem). This modelling language takes its origins in the GRAI methodology, for enterprise modelling and "decision centric" analysis. GRAI includes the original "GRAI Extended Actigram" modelling language (presented in section 2.4.1), for business processes. The language didn't possess an abstract syntax but several ones developed in the frame of academic researches and projects. In addition, the specification of GRAI Extended Actigram was not sufficiently formal.

The work performed on Extended Actigram Star consisted to re-engineer the original modelling language on a "meta model" architecture basis and to improve the usability of the language in the domain of Manufacturing Services (Model Driven Service Engineering Architecture) and its interoperability with other formalisms (e.g.: BPMN). Extended Actigram Star facilitates the modelling of business process in an enterprise offering a dynamic view of the process being modelled. It is addressed to business users responsible of the creation of the first model, business people responsible of the management, and to technical developers responsible of the development of business process modelling tools.

As a graphical modelling language, Extended Actigram Star will provide business users and analysts standards to visualize business processes in an enterprise, and thus with a comprehensible and easy way to handle these processes.

Thus, Extended Actigram Star **is a proposition of a new, more developed version of GRAI Extended Actigram**. It is based on a specific development strategy:

- Keep the core principles of GRAI Extended Actigram.
- Add new concepts in order to support abstraction, and to ease the implementation of software for model manipulation and transformation.

3.2 Overview

Business process is a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus's emphasis on what. A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action. Taking a process approach implies adopting the customer's point of view. Processes are the structure by which an organization does what is necessary to produce value for its customers [Davenport, 1993].

The world of business processes has changed dramatically over the past few years. Processes can be coordinated from behind, within and over organizations' natural boundaries. A business process now spans multiple participants and coordination can be complex.

Business process models can help business actors to handle the problems of heterogeneity, complexity, and flexibility in layered operational Enterprise Architectures and across the enterprise knowledge spaces of network life-cycles.

Extended Actigram Star language is suitable to collect knowledge about processes at the business level rather than other modelling language. This consideration is based on past experience of experts "from the field". This can be partially explained by its simplicity regarding other languages, which includes much more constructs and based on a more "IT Oriented" modelling approach.

3.3 Abstract Syntax

The diagram below is a Class diagram representation of the EA* Conceptual model, with its different sub packages, elements composing it, and their relations.

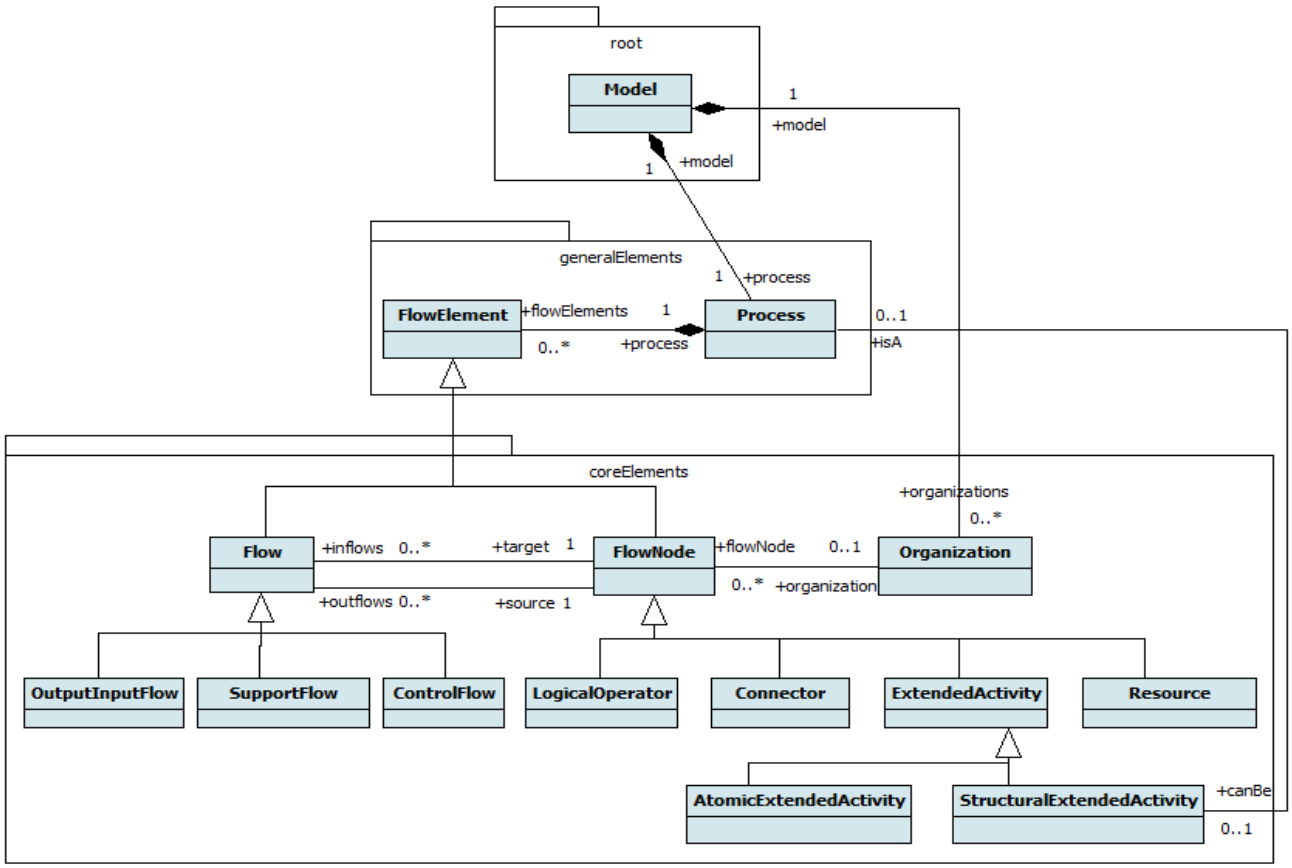


Figure 26 Abstract Syntax of Extended Actigram Star

3.3.1 **Structure**

Extended Actigram Star elements are divided into three sub packages:

- Package – Root: it contains the root element of the Extended Actigram star Language (Model)
- Package – General Elements: this package contains the generic classifiers of the language (“Flow Element” and “Process”) and factors out common attributes of the language constructs.
- Package – Core Concepts: this package contains the building blocks (“constructs”) of Extended Actigram Star models.

3.3.1.1 **Package: Root**

3.3.1.1.1 **Construct: Model**

Model is the root element of the Extended Actigram Star Conceptual model. It is composed of a process which is the subject to be modelled and might be composed of other processes belonging to the same domain of study.

Table 2 Model attributes

Class name	Model	
Inherits from	BaseElement (see BaseElement)	
Attribute	**Type**	**Description / Usage**
process	Reference	Is a reference to an object of type Process.
Organizations[0..*]	Reference	Set of organizations responsible for the realization of the process.

3.3.1.2 Package: General Elements

3.3.1.2.1 Abstract Construct: BaseElement

BaseElement is the most generic class of the Extended Actigram Star meta model. It is an abstract supper class, from which all other concepts inherit several common attributes.

Table 3 BaseElement attributes

Class name	BaseElement	
Inherits from	None	
Attribute	**Type**	**Description / Usage**
id	String	It is the unique id of an object; it is used for referencing each instance of this class.
name	String	The name of the object
code	String	to be identified
description	String	Object's description

The diagram below represents the BaseElement class, its attributes and the classes directly inheriting from it.

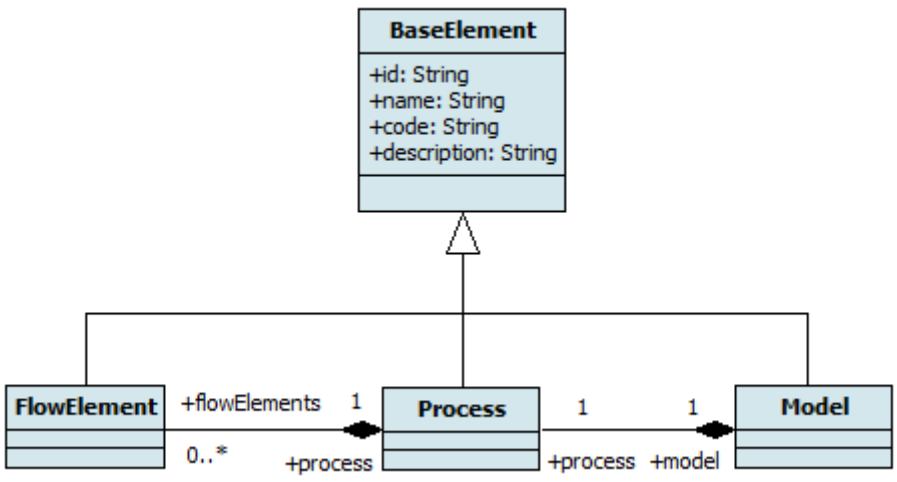


Figure 27 BaseElement

3.3.1.2.2 Construct: Process

Process is an essential concept of the language. It is a set of related, structural or atomic activities logically chained and triggered by flows and eventually using operators and connectors. Elements constituting a Process can be divided in two categories: nodes

(FlowNode) and flows (Flow). A Process is represented by one diagram. The following figure is a graph representation of a process and its decomposition.

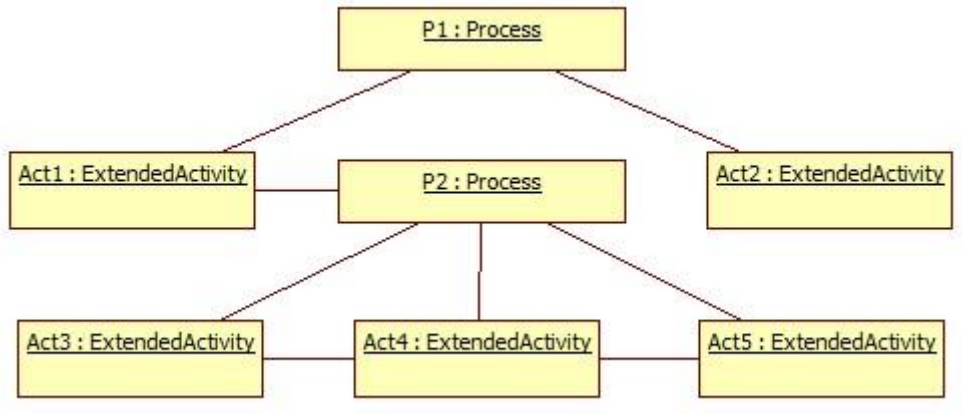


Figure 28 Process UML object diagram

The graph can be summarized in the following points:

- P1 represents the process to be modelled.
- P1 is composed of two ExtendedActivities: Act1 and Act2.
- Act1 is a structural ExtendedActivity connected to Process P2 ("isA" association relation).
- Act2 is an atomic ExtendedActivity, and can't be composed of other FlowElements.
- P2 is composed of three atomic ExtendedActivities: Act3, Act4, and Act5.

Table 4 Process attributes

Class name	Process	
Inherits from	BaseElement	
Attribute	**Type**	**Description / Usage**
FlowElements [0… *]	Reference	A Process is composed of FlowElements which represent all objects used to visualize a Process (ExtendedActivities, LogicalOperators, Resources and Connectors)

3.3.1.2.3 Abstract Construct: FlowElement

FlowElement is the core of Extended Actigram Star conceptual model. It emphasizes the notion of sequence within a process with all the conditions that govern this sequence.

FlowElement can be of two types:

- Flow: establish the connection between one node and another.
- FlowNode: are connectable elements which can be linked to one another by means of a Flow. Thus, a FlowNode can be a source or target of a flow.

FlowElement class is a generic class of all the objects that constitute a process. As a result, every object which appears in a process diagram is a FlowElement.

Table 5 FlowElement attributes

Class name	FlowElement	
Inherits from	BaseElement	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3 Core Elements

3.3.1.3.1 Abstract Construct: Flow

A Flow is the link connecting two FlowNodes; it represents the exchange of objects (information, products, resources, etc.) between ExtendedActivities, LogicalOperators, Connectors and Resources. Each instance is characterized by a "Source" and a "Target". Besides, a flow is able to activate or initiate an ExtendedActivity, depending on the value assigned to "isTrigger" attribute.

Table 6 Flow attributes

Class Name	Flow	
Inherits from	FlowElement	
Attribute	**Type**	**Description / Usage**
isTrigger	Boolean	The triggering characteristic determines if a flow is capable to trigger an activity or not. It is of a Boolean type
triggerInfo	String	to be identified
source	Reference	The source of the Flow
Target	Reference	The target of the Flow

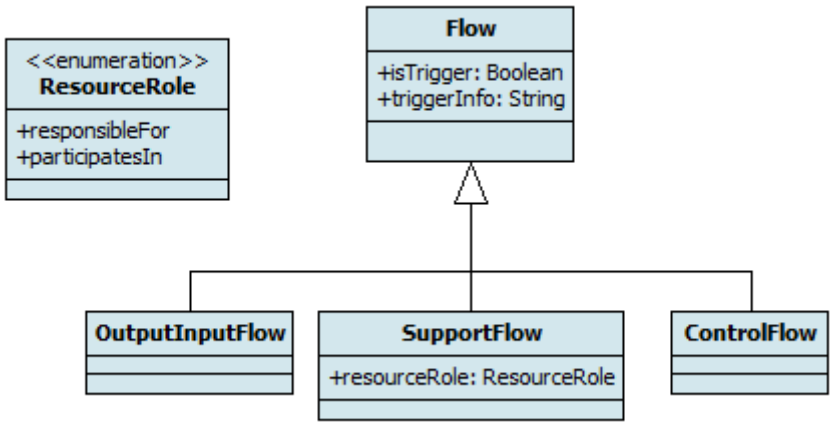


Figure 29 Flow

3.3.1.3.2 Construct: OutputInputFlow

An OutputInputFlow depicts the logical sequence between two elements. It inherits its attributes from the Flow class.

Table 7 OutputInputFlow attributes

Class Name	OutputInputFlow	
Inherits from	FlowElement	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.3 Construct: ControlFlow

A ControlFlow describes the conditions or rules that govern the execution of an ExtendedActivity. It inherits its attributes from the Flow class.

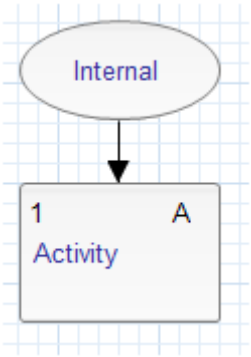


Figure 30 ControlFlow

In Figure 30 the InternalConnector "design office" sends designs to the "cutting shirts" structural ExtendedActivity. These designs describe the rules of how the shirts should be cut in the "cutting shirts". The flow in figure 30 is a ControlFlow.

Table 8 ControlFlow attributes

Class Name	ControlFlow	
Inherits from	FlowElement	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.4 Construct: SupportFlow

A SupportFlow indicates that the flow is supporting the realization of an activity or of a process. Each instance of the SupportFlow whose source is not a Material Resource, has a "resourceRole" which can be:

- responsible for: The IT or Human resource (source of the SupportFlow) is responsible for the supported ExtendedActivity. It represents a general role like manager or customer in case of Human resources, or information system in case of IT resource

- participates in: The resource (source of the SupportFlow) participates in the execution of the supported ExtendedActivity without being responsible for it.

A SupportFlow whose source is a Material Resource has only "participates in" as resourceRole. Several resources can support the same ExtendedActivity, the SupportFlows connecting these resources and the target ExtendedActivity should obey a single constraint: Only one of these SupportFlows can possess a resourceRole whose value is "responsible for". This constraint is due to the fact that only one resource (IT or Human) can be responsible for a resource while the others would be participants.

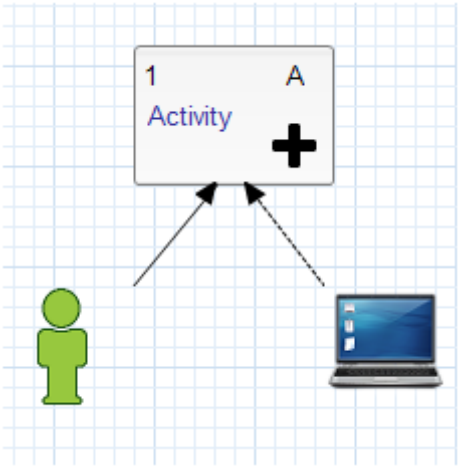


Figure 31 SupportFlow Example

Figure 31 is an Example of Process composed of:

- Structural ExtendedActivity named "customer's login" which manages the login of customers into an online marketplace.
- Human resource named "customer" who is responsible for the execution of the "customer's login"
- IT resource named "IT system" which participates in the execution of the "customer's login"

Table 9 SupportFlow attributes

Class Name	SupportFlow	
Inherits from	FlowElement	
Attribute	**Type**	**Description / Usage**
resourceRole	Enumeration	"responsible for" or "participates in"

3.3.1.3.5 Abstract Construct: FlowNode

A FlowNode is a generic concept which defines one of the 4 basic elements that compose a Process:

- ExtendedActivity
- LogicalOperator
- Connector
- Resource

FlowNodes are regarded to be the building blocks of a Process. These blocks are connected using Flows (Flow). FlowNodes can be target or source of a Flow. Connections between FlowNodes are governed by a set of constraints (Flow constraints) depending on its types

Table 10 FlowNode attributes

Class name	FlowNode
Inherits from	FlowElement

Attribute	Type	Description / Usage
none		

3.3.1.3.6 Abstract Construct: ExtendedActivity

ExtendedActivity represents the functional unit of a Process. Structural ExtendedActivities can be decomposed into other ExtendedActivities while atomic ones can't. An ExtendedActivity can start/end the process execution. A process can have several starting/ending Extended Activities in case of parallel execution.

Table 11 ExtendedActivity attributes

Class name	ExtendedActivity	
Inherits form	FlowNode, Process	
Attribute	**Type**	**Description / Usage**
isStarting	Boolean	A Boolean value indicating if the ExtendedActivity is a starting Activity or not
isEnding	Boolean	A Boolean value indicating if the ExtendedActivity is an ending Activity or not
mission	String	to be identified
functionalRules	String	to be identified
minCost	Double	to be identified
maxCost	Double	to be identified
averageCost	Double	to be identified
minTimeDelay	Double	to be identified
maxTimeDelay	Double	to be identified
averageTimeDelay	Double	to be identified

Some attributes are marked as "to be identified" since their exact usage is not still identified. These attributes are implemented but not used while modelling a process until the moment and their usage will be identified with respect to the process's simulation requirements.

The diagram below represents the ExtendedActivity class and its relations.

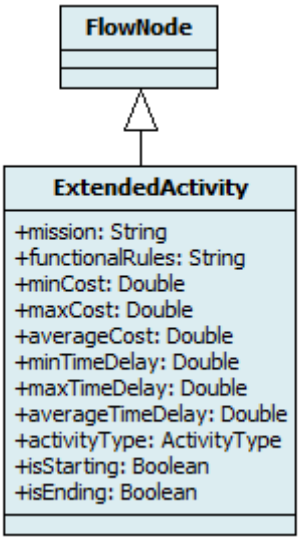


Figure 32 ExtendedActivity

3.3.1.3.7 AtomicExtendedActivity

AtomicExtendedActivity is an ExtendedActivity which is not decomposed into other flowElements. It inherits its attributes from the ExtendedActivity abstract class.

Table 12 AtomicExtendedActivity attributes

Class name	AtomicExtendedActivity	
Inherits from	ExtendedActivity	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.8 StructuralExtendedActivity

StructurralExtendedActivity is an ExtendedActivity which can be decomposed into other flowElements. In addition to attributes inherited from the ExtendedActivity abstract class it possesses a reference to a Process.

Table 13 StructuralExtendedActivity attributes

Class name	StructuralExtendedActivity	
Inherits from	ExtendedActivity	
Attribute	**Type**	**Description / Usage**
isA	Reference	A StructuralExtendedActivity is a Process which in its turn contains flowElements that compose the activity

3.3.1.3.9 Abstract Construct: Resource

Resource represents all kinds of resources and used by a process during transformation or which has played a role in the process execution. Resource can be of three kinds:

- Human: a human who participated in the execution of an ExtendedActivity by delivering his competences or taking a role in the execution.
- Material: a material used by an ExtendedActivity such as camions, machines…
- IT: a computer software playing a role in the execution of an ExtendedActivity

A Resource is an abstract class which extends the FlowNode class.

Table 14 Resource attributes

Class name	Resource	
Inherits from	FlowNode	
Attribute	**Type**	**Description / Usage**
UnitaryCost	Double	to be identified
location	String	to be identified
capabilities	String	to be identified

The following diagram represents the Resource abstract class and its three types.

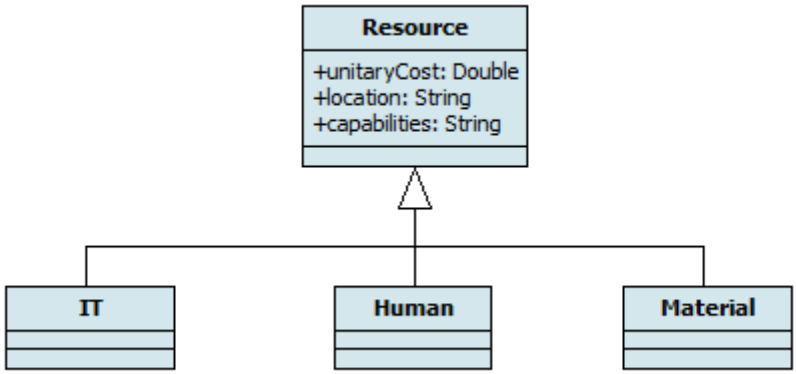


Figure 33 Resource

3.3.1.3.10 Construct: Human

Human resources are humans who support the process execution or responsible for a process.

Table 15 Human attributes

Class name	Human	
Inherits from	Resource	
Attribute	**Type**	**Description / Usage**
none		

An **example** of a Human resource is an employee called David whose role is to use a scanner in an ExtendedActivity responsible for scanning materials.

3.3.1.3.11 Construct: Material

Material resource represents technical resources and machines involved in the process execution.

Table 16 Material attributes

Class name	Material	
Inherits from	Resource	
Attribute	**Type**	**Description / Usage**
none		

An example of a Material resource is the scanner used in the previous example.

3.3.1.3.12 Construct: IT

IT resources represent all computer software playing a role in the execution of a process

Table 17 IT attributes

Class name	IT	
Inherits from	Resource	
Attribute	**Type**	**Description / Usage**
none		

An example of an IT resource is software used by an employee to manage the storage of materials in a depot.

3.3.1.3.13 Abstract Construct: LogicalOperator

LogicalOperators are used to control how the Process flows through different types of Flows as they converge and diverge within a Process. If the flow does not need to be controlled, then a LogicalOperator is not needed. LogicalOperators allow or disallow passage of flows which can be merged together on input or split apart on output.

Table 18 LogicalOperator attributes

Class name	LogicalOperator	
Inherits from	FlowNode	
Attribute	**Type**	**Description / Usage**
none		

The following diagram represents LogicalOperator and its types.

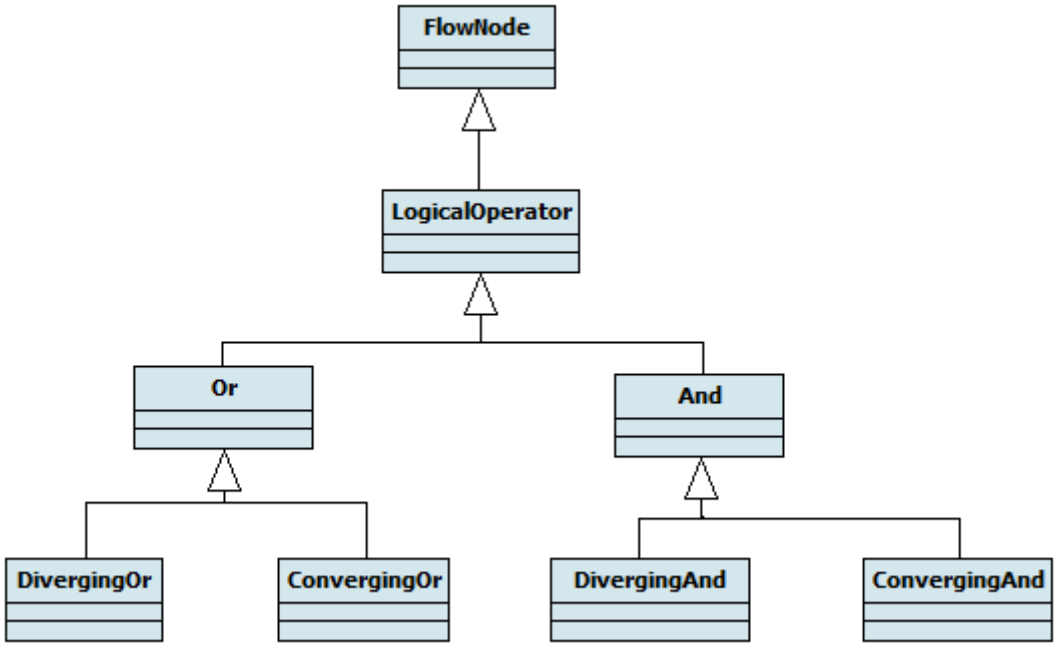


Figure 34 LogicalOperator

3.3.1.3.14 Construct And

The And LogicalOperator is a generic class for DivergingAnd and ConvergingAnd classes.

Table 19 Diverging attributes

Class name	And	
Inherits from	LogicalOperator	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.15 Construct: DivergingAnd

DivergingAnd takes one flow as input and has at least two flows as output. The output paths will all start at different or similar time.

Table 20 DivergingAnd attributes

Class name	DivergingAnd	
Inherits from	And	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.16 Construct: ConvergingAnd

ConvergingAnd has at least two input flows and it has one output flow. All input flows should terminate before the output flow continues.

Table 21 ConvergingAnd attributes

Class name	ConvergingAnd	
Inherits from	Converging	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.17 Construct Or

The Or LogicalOperator is a generic class for ConvergingOr and DivergingOr classes.

Table 22 Converging attributes

Class name	Or	
Inherits from	LogicalOperator	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.18 Construct: DivergingOr

DivergingOR allows one and only one input flow and has at least two output flows. Only one of its output flows can start execution.

Table 23 DivergingOr attributes

Class name	DivergingOr	
Inherits from	Or	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.19 Construct: ConvergingOr

ConvergingOr has at least two input flows and it has one output flow. Any of the input flows have to be terminated before the output flow continues.

Table 24 ConvergingOr attributes

Class name	ConvergingOr	
Inherits from	Converging	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.20 Abstract Construct: Connector

Connectors are used to represent the origin or the destination of a flow when the origin or the destination is outside the current diagram. Possible types are: ProcessConnector, InternalConnector, or ExternalConnector.

In order to provide a clear definition of Connectors and differentiate between its types the "domain of study" should be clearly defined

Domain of study represents the borders of the process to be modelled.

- If a target or source of a flow is outside the domain of study it is regarded as an ExternalConnector.
- If a target or a source of a flow belongs to the domain of study but it is not modelled since it is not of a great interest to the modeler, it is regarded as InternalConnector.
- If a target or a source of a flow belongs to the domain of study but it is modelled, it is regarded as a ProcessConnector.

The following diagram represents the Connector class and its three types.

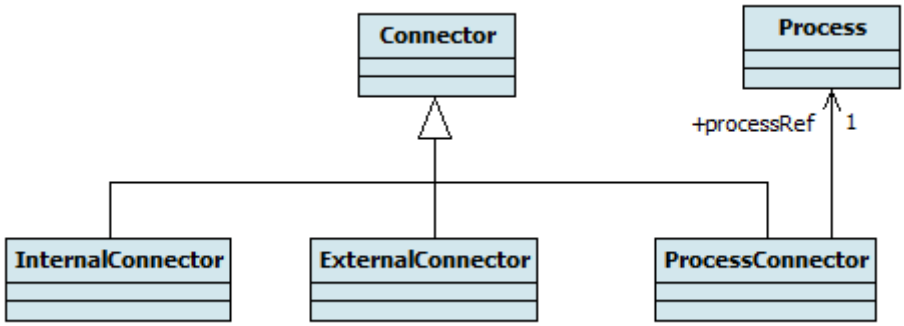


Figure 35 Connector

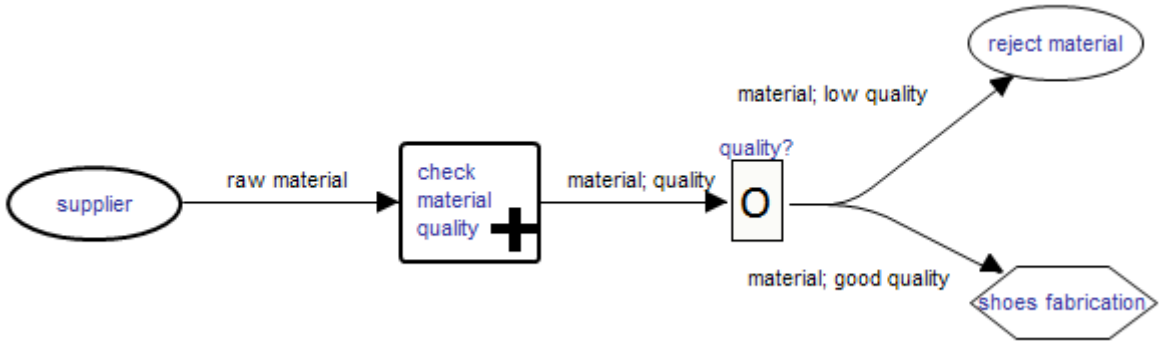


Figure 36 check material quality example

Figure 36 is an EA* diagram represents a process of checking the quality of all imported raw materials from supplier in a shoes fabrication industry. The supplier supplies the industry with raw material (leather and rubber), and then the quality of these materials is checked (good or low quality). If the material's quality is low, it would be rejected. Else it would be send to the shoes fabrication process.

The domain of study of this process is the shoes production starting from checking material's quality, to designs, fabrication, delivery etc…

3.3.1.3.21 Construct: InternalConnector

The InternalConnector indicates that the origin or the destination belongs to the domain of study but it is not modelled. In Figure 36, the "reject material" is an InternalConnector which represents a process whose role is to reject all low quality materials. The process belongs to the domain of but it is not modelled yet.

Table 25 InternalConnector attributes

Class name	InternalConnector	
Inherits from	Connector	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.22 Construct: ExternalConnector

The ExternalConnector indicates that the origin or the destination doesn't belong to the domain of study and it might be modelled or not. In figure 36 the "supplier" is represented by an ExternalConnector since it doesn't belong to the domain of study.

Table 26 ExternalConnector attributes

Class name	ExternalConnector	
Inherits from	Connector	
Attribute	**Type**	**Description / Usage**
none		

3.3.1.3.23 Construct: ProcessConnector

The ProcessConnector indicates that the origin or the destination belongs to the domain of study and that it is modelled. In figure 36 the "shoes fabrication" process is represented by a ProcessConnector since it belongs to the domain of study and modelled.

Table 27 ProcessConnector attributes

Class name	ProcessConnector	
Inherits from	Connector	
Attribute	**Type**	**Description / Usage**
processReference	Reference	A reference to the Process referenced by this connector

3.3.1.3.24 Construct: Organization

Organization permits to represent organizations participating in the realization of a process. An organization is responsible for specific FlowNodes in a process.

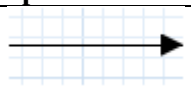
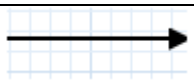
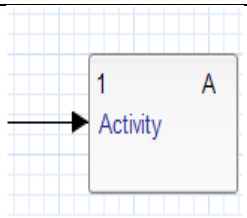
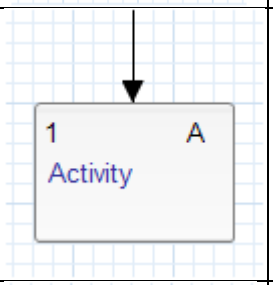
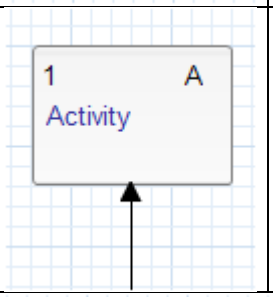
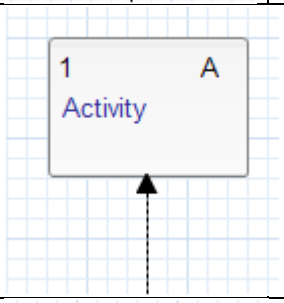
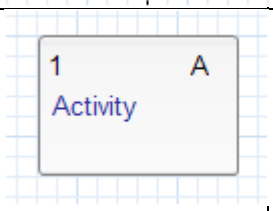
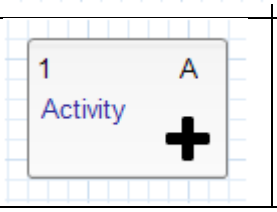
Table 28 Organization attributes

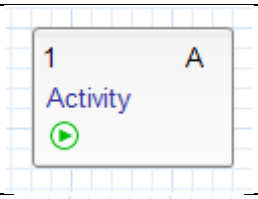
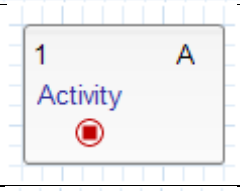
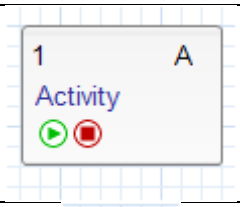
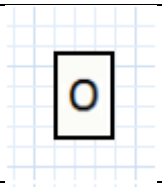
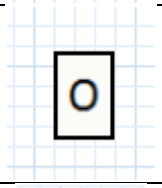
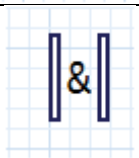
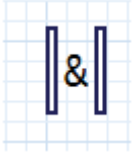
Class name	Organization	
Inherits from	BaseElement	
Attribute	**Type**	**Description / Usage**
flowNode[0..*]	Reference	Set of FlowNodes which the organization is responsible for

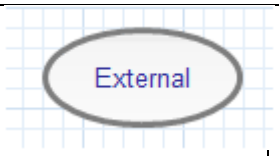
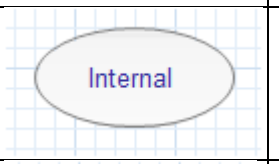
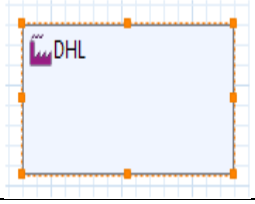
3.4 Graphical Representations and Notations

This section is made up of a table that summarizes the mapping between Extended Actigram Metamodel classes and the graphical representations of the GRAI Extended Actigram.

Table 29 Graphical Representations

Concept	Condition	Graphical Notation	Description
Flow	Attribute "isTrigger" is set to false		It is a normal arrow oriented from source to target
	Attribute "isTrigger" is set to true		It is a thick arrow oriented from source to target
OutputInputFlow	It's target/source is an ExtendedActivity		The arrow can be connected to the ExtendedActivity at its right/left corners
ControlFlow	It's target is an ExtendedActivity		The arrow can be connected to the ExtendedActivity at its upper corner
SupportFlow	Attribute resourceRole is set to "responsible for"		The arrow can be connected to the ExtendedActivity at its bottom corner
	Attribute resourceRole is set to "participates in"		The dashed arrow can be connected to the ExtendedActivity at is bottom corner
ExtendedActivity	AtomicExtendedActivity		A rectangle with the name of the activity in the middle
	StructuralExtendedActivity		A rectangle with the name of the activity in the middle and plus sign at the right bottom.

	Attribute "isStarting" is set to true		A green circle located at the left bottom corner of the activity
	Attribute "isEnding" is set to true		A red circle located at the left bottom corner of the activity
	Attributes "isStarting" and "isEnding" are set to true		A green and red circles located at the left bottom corner of the activity
Resource	Human		
	Material		
	IT		
Logical Operator	DivergingOr		The "O" represents the OR feature. The peak to the left represents divergence
	ConvergingOr		The "O" represents the OR feature. The peak to the right represents convergence
	DivergingAnd		The "&" represents the AND feature. The peak to the left represents divergence
	ConvergingAnd		The "&" represents the AND feature. The peak to the right represents convergence

Connector	ExternalConnector		A circle with thick border indicating that it is external and with the name or a description of the referenced process in its middle.
	InternalConnector		A circle with the name or a description of the referenced process in its middle
	ProcessConnector		A hexagon with the name or a description of the referenced process
Organization			

3.5 Connectivity constraints

Well-Formedness (or static semantics) defines the rules (constraints) that govern relations between classes. In Extended Actigram Star language, several rules and constraints are defined which govern the relations and connections between different FlowElements. In order to represent these constraints:

- Textual annotations can be associated with the UML metamodel at design level.
- OCL can be used at design and implementation level.

The following table summarizes the rules that apply to the utilization of Flow, depending on the target and the source objects that are to be connected.

Table 30 Flow Constraints

	Target				
		ExtendedActivity	**LogicalOperator**	**Resource**	**Connector**
Source	**Extended Activity**	OutputInputFlow(trigger) ControlFlow (trigger) SupportFlow	OutputInputFlow	N.A.	OutputInputFlow
	Logical Operator	OutputInputFlow(trigger) ControlFlow (trigger)	OutputInputFlow	N.A.	OutputInputFlow
	Resource	SupportFlow	N.A.	N.A.	N.A.
	Connector	OutputInputFlow (trigger) ControlFlow (trigger)	OutputInputFlow	N.A.	N.A.

N.A. = Not Applicable

(trigger) = is an optional characteristic of a flow.

Triggering characteristic

A Flow can trigger an ExtendedActivity, but with constraints depending on its source, target, and flowRole:

- The target of a flow **must** be an ExtendedActivity.
- The flow **must** be OutputInput or Control.
- A support flow cannot be a triggering flow.

4. Model Transformation

The MDA guide [OMG, 2003] defines a model transformation as "the process of converting one model to another model of the same system". [Kleppe et al, 2003] defines a transformation as the automatic generation of a target model from a source model, according to a transformation definition. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed to a model in the target language. A transformation rule is a description of how one or more constructs in the source language can be transformed to one or more constructs in the target language. The aim of model transformation is to carry out automated translations within and between modeling languages. Model Driven Service Engineering Architecture (MDSEA) regards model transformation as an essential aspect for accomplishing interoperability. It defines a framework for model transformations based on vertical and horizontal transformations. Several transformations had been identified, specified and implemented in the SLMToolBox modeling tool. These transformations insured the automatic passage from one abstraction level to another (MDSEA abstraction levels) and assisted the simulation of business processes. MDSEA define the modeling languages to use at each abstraction level. Two business process modeling languages have been identified: Extended Actigram Star (EA*) originating from GRAI Extended Actigram [add reference] and BPMN [add reference]. These two modeling languages are used at two different separated abstraction levels. The need of a model transformation is crucial, in order to transform EA* models developed at one abstraction level into BPMN models at another level.

4.1 Problem

ASICOM was a French funded project, whose goal was to build a platform that enables interoperability among industrial partners. Model transformation was a key solution to interoperability issues. In the frame of this project, transformations from GRAI Extended Actigram models to UML activity diagrams and BPMN models [OMG-2, 2011] were tested and evaluated. The ASICOM team has encountered several problems during their research, based on the current GRAI Extended Actigram language version which was not designed within a MDA approach and thus imposes limits on the transformation of models generated by this language. It did not possess an official MOF metamodel, but several metamodels developed in the frame of academic researches and projects. In addition, as explained earlier the specification of GRAI Extended Actigram was not sufficiently formal to allow the transformation into other formalisms. In this section we present the transformation from EA* (the new developed version of GRAI Extended Actigram) models to BPMN models.

On the other side, MDSEA starts at the strategic level of companies that want to evolve towards service-oriented business methods. The "to-be" business specific model is specified and developed at the BSM level. Later, detailed functional definition model is developed at

the TIM level, and a practical implementation model at TSM level. The passage from one High level to another lower level should be implemented at the basis of model reuse and enrichment. One major problem that is frequently identified in the enterprises is the gap between people visions to describe the process. The process can be either defined for creating physical product or services. It opposes on one side the business view and on the other the technical one. Some efforts between have been produced to reduce this gap. BSM models vary from TIM models due to the different modelling languages used at each level. Business processes at BSM level are modelled using the Extended Actigram Star language while at TIM level business processes are modelled using Business process modelling notation (BPMN). These two different modelling languages are based on different metamodels, and thus different specific constructs. This reveals a need to transform BSM models into TIM models and in its turn TIM models into TSM ones.

4.2 MetaModel Approach

The objective is to transform BSM source models into a TIM target Models. One of the most used transformation techniques is the "Metamodel Approach" [Bourey JP. 2007]. Figure 37 particularizes the "Metamodel" approach to the context of transformation of EA* models into BPMN2.0 models.

Step 1 of the transformation, and a mandatory pre-requisite, is the formalization of the source and target meta-models (respectively EA* metamodel and BPMN2.0 metamodel). Ecore (which is part of Eclipse EMF) is an implementation of a simplified form of MOF [OMG 2006] and is used to define source ant target metamodels. In addition XML Metadata Interchange (XMI) [OMG 2000] is used to save source and target models.

Step 2 of the transformations methodology marks the beginning of the actual design of the transformations by defining the model mappings that relate the concepts of each meta-model. From a syntactic point of view, the mapping is a morphism that must ensure the consistency of source and target models, and 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) while leaving both intact. In transformations, the source model is transformed using a function that applies a mapping to produce a different target model. This function can be expressed either explicitly, using graphs, sets, tuples, or even mapping tables relating multiple or single constructs and stored in a physical location; or implicitly in the developer's mind. However, in both cases is necessary to implement them using a transformation language (step 3).

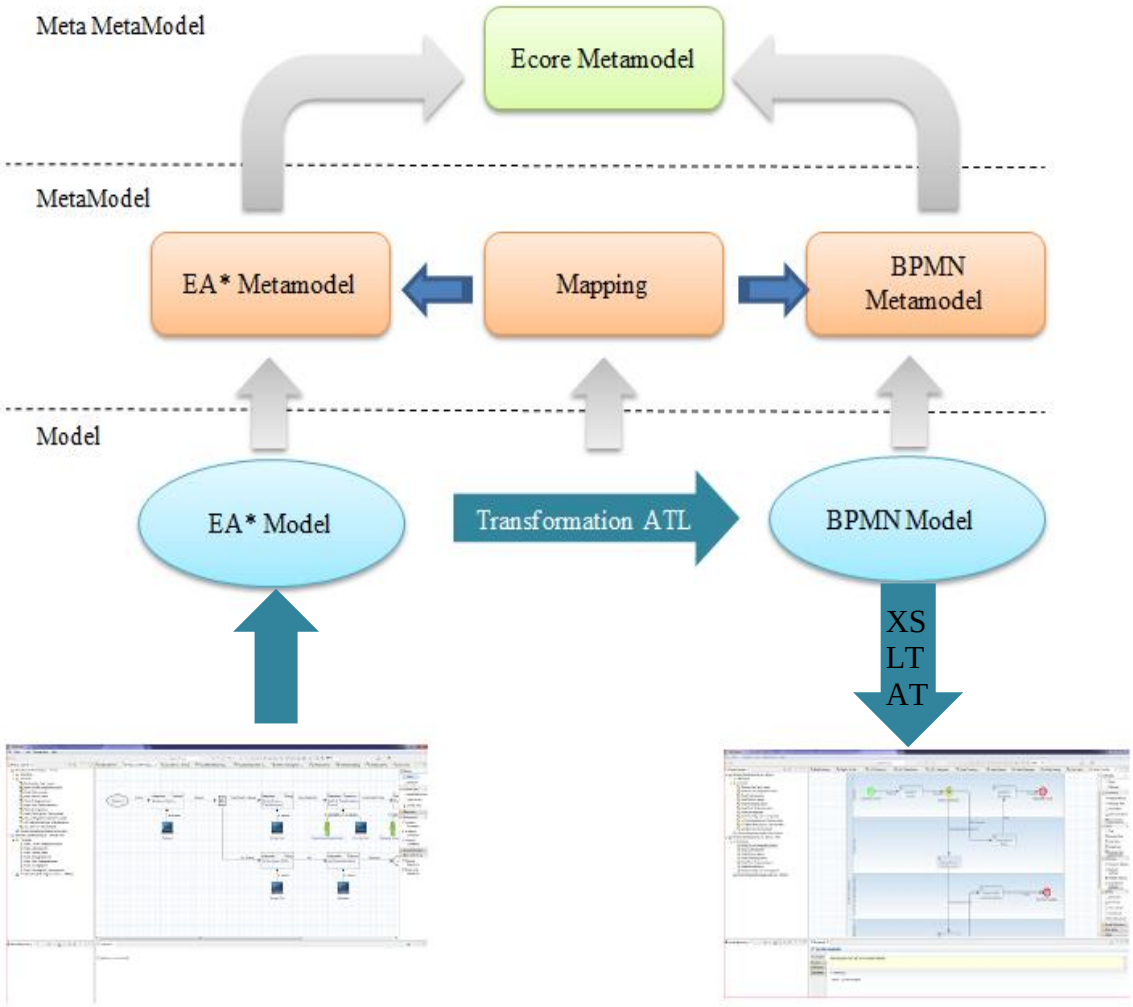


Figure 37 Transformation architecture of EA* to BPMN

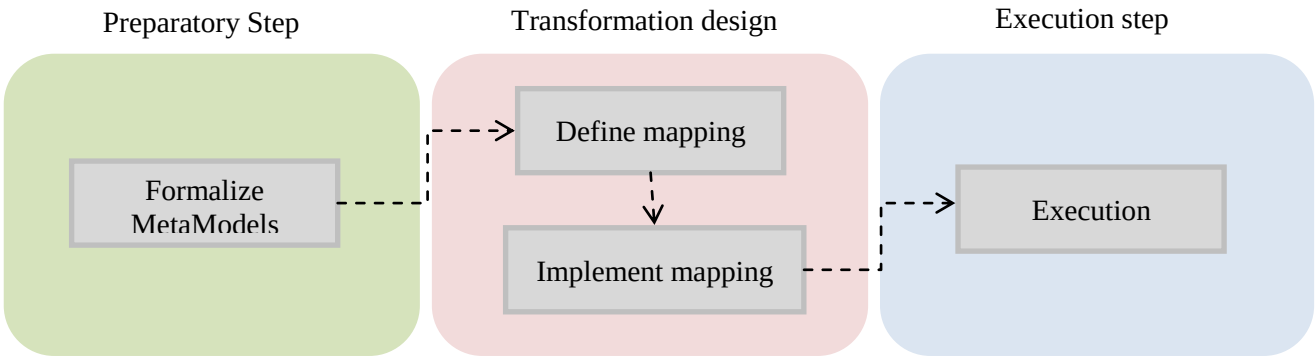


Figure 38 Different steps for transformation

Model transformation is an important activity in Model-Driven Engineering, and OMG recognized this by issuing the Query/Views/Transformations (QVT) request for proposals to seek an answer compatible with its MDA standard suite. Many contributions 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), like QVT itself and ATL [Jouault et al, 2008], which despite not being a standard is one of the most used, having a large user's base. Nevertheless, as enumerated in [Czarnecki & Helsen 2006], others can also be used and applied to the implementation of the model mappings, e.g. Xtend/Xpand, UMLX, AToM3, MTL, etc. As analyzed by [Agostinho 2011], some of the above languages are ideal for the representation structural mappings, others for semantic maps, providing good human traceability, while others are more formal and mathematical based. However, none provides the capability or the APIs to translate explicit mappings into executable code. Mappings implemented with them are normally static and any change obliges to manually rewrite code. Benefiting from a good JAVA integration that enables to address the above problems in the future and having a considerable amount of support through online communities, ATL has been the elected language for MDSEA mappings implementation.

4.3 Mapping of Concepts

The mapping of concepts proposed for the transformation creates correspondences and links between concepts and their relations from EA* to BPMN language. It is a translation of constructs and their relations from one metamodel to another. As a result, deep analysis and understanding of the EA* and BPMN metamodels, represent the main key to start in translation and drawing the links. Investigating the concepts and frame of transformation from EA* to BPMN models resulted in two different types of mapping. The first type is more concerned in transformation within the frame of MDSEA methodology, while the second is a more general transformation that passes the borders and limits of MDSEA. This section presents the two kind of results obtained (conceptual).

4.3.1 **Results in the frame of MDSEA**

In MDSEA the mapping of concepts is constrained by the modelling rules of EA* at the BSM level. BSM is composed into:

- BSM Top level: a general view of the system to be modelled.
- BSM bottom level: a decomposed and more detailed view.

Modelling a business process at BSM level started at the top level with a general view of the process, and then this process is decomposed into several more detailed processes. This modelling strategy takes the form of a pyramid as explained in figure 39. Modelling starts from top to bottom, a first EA* process representing general view of the business process is modelled. Later this general view is decomposed into several more detailed processes.

Figure 40 represents the applied scenario in order to transform an EA* diagram into a BPMN diagram in the frame of MDSEA. The business process is first modeled at the BSM top level from a general view. Then the resulted EA* diagram is manually decomposed at the BSM bottom into two separate diagrams. These two diagrams are later transformed using the "EA* to BPMN2.0 model transformation" into a BPMN diagram each. The resulted BPMN diagrams are then regrouped together manually by the user at TIM top level.

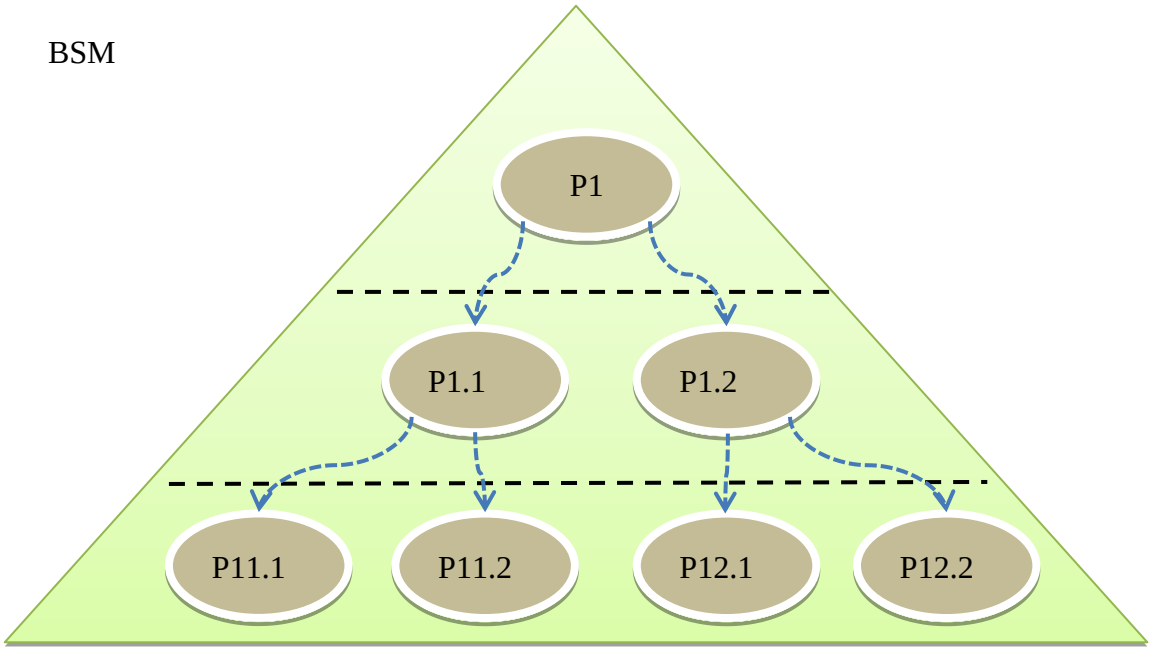


Figure 39 BSM Modelling strategy

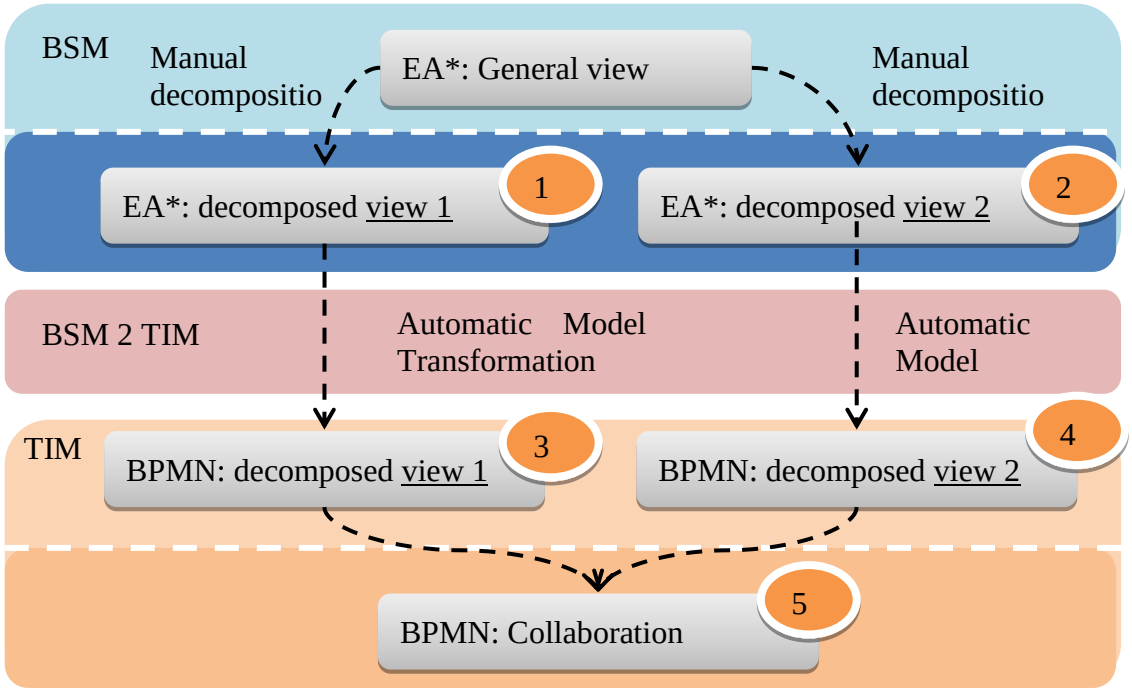


Figure 40 EA* to BPMN collaboration

4.3.1.1 EA* to BPMN Collaboration Diagram – Mapping

The following table summarizes the mapping of EA* concepts to BPMN concepts. The mapping is accompanied with conditions which governs the creation of relations.

Table 31 EA* to BPMN - Mapping (Collaboration Diagram)

EA*	Condition	BPMN2.0
Model		Definitions
Process		Process, Participant
ExtendedActivity	Structural	Sub Process
	Atomic & supported by Human	UserTask
	Atomic & supported by IT (no human interaction)	ServiceTask
DivergingOr		Diverging Exclusive Gateway
ConvergingOr		Converging Exclusive Gateway
DivergingAnd		Parallel Gateway
ConvergingAnd		Parallel Gateway
MaterialResource	Material	Data Object
HumanResource	Responsible for	Lane
	Participates in	Resource (added to the list of resources of a task)
ITResource	Responsible for	Lane
	Participates in	Resource (added to the list of resources of a task)
Organization		Lane
Control Flow	If the source is an ExternalConnector or InternalConnector and target is an “atomic” ExtendedActivity	MessageFlow
	If the source is an ExternalConnector or InternalConnector and target is a “structural” ExtendedActivity	Catching Message Event, Message flow, and Sequence Flow
	If the source is a ProcessConnector or ExtendedActivity	DataObject, and associations
OutputInputFlow	If the source is an ExternalConnector or InternalConnector (and target is an atomic ExtendedActivity)	MessageFlow
	If the source is an ExternalConnector or InternalConnector (and target is a structural ExtendedActivity or LogicalOperator)	Catching Message Event, Message Flow, and Sequence Flow
	If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity or ProcessConnector or logical operator)	SequenceFlow
	If the source is a structural ExtendedActivity or logical operator (and target is an ExternalConnector or InternalConnector)	Throwing Message Event, Message Flow, Sequence Flow
	If the source is an atomic ExtendedActivity (and target is an External or InternalConnector)	MessageFlow

SupportFlow	If source is a material resource	Association
ExternalConnector		Participant (Pool)
ProcessConnector		Call Activity
InternalConnector		Participant (Pool) (Black Box)

4.3.1.2 EA* to BPMN Collaboration Diagram – Transformation Rules

Atomic ExtendedActivity

- A Human resource is responsible for the realization of the ExtendedActivity. In this case the atomic ExtendedActivity is mapped to a UserTask.
- An IT resource is responsible for the realization of the ExtendedActivity. In this case the atomic ExtendedActivity is mapped to a ServiceTask.

Resource

- The value of the resourceRole is "responsible for". In this case the resource (Human or IT) is mapped to a lane, in which the supported ExtendedActivity belongs to the lane.
- The value of the resourceRole is "participates in". In this case the resource (Human or IT) is added to the list of resources to the supported ExtendedActivity.

ControlFlow

- Source is an ExternalConnector or InternalConnector and target is an "atomic" ExtendedActivity. In this case it is mapped to a MessageFlow.
- Source is an ExternalConnector or InternalConnector and target is a "structural" ExtendedActivity. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.
- Source is a ProcessConnector or ExtendedActivity. It is a "1 to n relation", in which the "Control" Flow is mapped to a combination of DataObject and two Associations.

OutputInputFlow

- Source is an ExternalConnector or InternalConnector and target is an atomic ExtendedActivity. In this case the "OutputInput" Flow is mapped to a MessageFlow**.**
- Source is an ExternalConnector or InternalConnector and target is a structural ExtendedActivity or LogicalOperator. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.
- Source is a ProcessConnector, ExtendedActivity, or LogicalOperator and target is also one of these three options. In this case it is mapped to SequenceFlow.
- Source is a structural ExtendedActivity or LogicalOperator, and target is an ExternalConnector or InternalConnector. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow,

throwing MessageEvent, and a SequenceFlow.

- Source is an atomic ExtendedActivity and target is an ExternalConnector or InternalConnector. In this case it is mapped to a MessageFlow.

SupportFlow

- Source is a Material resource. In this case it is mapped to an Association**.**

4.3.1.3 EA* to BPMN Process Diagram – Model Mapping

The following table summarizes the mapping of EA* concepts to BPMN concepts. The mapping is accompanied with conditions which governs the creation of relations.

Table 32 EA* to BPMN - Mapping (Collaboration Diagram)

EA*	Condition	BPMN2.0
Model		Definitions
Process		Process
ExtendedActivity	Structural	Sub Process
	Atomic & supported by Human	UserTask
	Atomic & supported by IT (no human interaction)	ServiceTask
DivergingOr		Diverging Exclusive Gateway
ConvergingOr		Converging Exclusive Gateway
DivergingAnd		Parallel Gateway
ConvergingAnd		Parallel Gateway
MaterialResource	Material	Data Object
HumanResource	Responsible for	Lane
	Participates in	Resource (added to the list of resources of a task)
ITResource	Responsible for	Lane
	Participates in	Resource (added to the list of resources of a task)
Organization		Lane
Control Flow	If the source is a ProcessConnector or ExtendedActivity	DataObject, and associations
OutputInputFlow	If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity or ProcessConnector or logical operator)	SequenceFlow
SupportFlow	If source is a material resource	Association
ExternalConnector		Not mapped
ProcessConnector		Call Activity
InternalConnector		Not mapped

4.3.1.4 EA* to BPMN Process Diagram – Transformation Rules

Process

- An EA* Process is mapped to a BPMN2.0 Process, this Process won't be represented graphically by a pool (in contrast to Collaboration BPMN2.0 diagrams)

ControlFlow

- if the source is a ProcessConnector or ExtendedActivity (Atomic or Structural), then it is mapped to a sequence flow
- if condition 1 is not applicable then a ControlFlow is not mapped

OutputInputFlow

- Source is a ProcessConnector, ExtendedActivity, or LogicalOperator and target is also one of these three options. In this case it is mapped to SequenceFlow
- if condition 1 is not applicable then an OutputInputFlow is not mapped

SupportFlow

- if source is a material resource, then it is mapped to an Association
- if source is of another type, then it is not mapped

Connector

- If the connector is a ProcessConnector then it is mapped to a CallActivity. Any other type of connectors is not mapped

4.3.2 Results outside the frame of MDSEA (Generalization)

In the previous section we presented the mapping and transformation rules of EA* model into BPMN process and collaboration models. These mappings and rules were the work results in the frame of MDSEA and thus coherent with the MDSEA modeling methodology explained earlier. In this section we present a mapping and transformation rules from EA* to BPMN regardless of the MDSEA rules and method in an attempt for a general transformation and not limited to MDSEA.

4.3.2.1 EA* to BPMN Collaboration Diagram – Mapping

The following table summarizes the mapping of EA* concepts to BPMN concepts. The mapping is accompanied with conditions which governs the creation of relations.

Table 33 EA* to BPMN - Mapping (Collaboration Diagram)

EA*	Condition	BPMN2.0
Model		Definitions
Process		Process, Participant(Pool)
ExtendedActivity	Structural	Sub Process
	Atomic & supported by Human	UserTask
	Atomic & supported by IT (no human interaction)	ServiceTask
DivergingOr		Diverging Exclusive Gateway

ConvergingOr		Converging Exclusive Gateway
DivergingAnd		Parallel Gateway
ConvergingAnd		Parallel Gateway
MaterialResource	Material	Data Object
HumanResource		Resource (added to the list of resources of a task)
ITResource		Resource (added to the list of resources of a task)
Organization		Process, Participant(Pool)
Control Flow	If the source is an ExternalConnector or InternalConnector and target is an "atomic" ExtendedActivity	MessageFlow
	If the source is an ExternalConnector or InternalConnector and target is a "structural" ExtendedActivity	Catching Message Event, Message flow, and Sequence Flow
	If the source is a ProcessConnector or ExtendedActivity	DataObject, and associations
OutputInputFlow	If the source is an ExternalConnector or InternalConnector (and target is an atomic ExtendedActivity)	MessageFlow
	If the source and target are of type "atomic" ExtendedActivity and don't belong to the same organization	MessageFlow
	If the source is an atomic ExtendedActivity (and target is an External or InternalConnector)	MessageFlow
	If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is an ExtendedActivity or ProcessConnector or logical operator) and source and target belong to the same organization	SequenceFlow
	If the source is an ExtendedActivity, ProcessConnector, or LogicalOperator (and target is an ExtendedActivity or ProcessConnector or logical operator) and source and target don't belong to the same organization	Throwing Message Event, Message Flow, Sequence Flow
	If the source is a structural ExtendedActivity or logical operator (and target is an ExternalConnector or InternalConnector)	Throwing Message Event, Message Flow, Sequence Flow
	If the source is an ExternalConnector or InternalConnector (and target is a structural ExtendedActivity or LogicalOperator)	Catching Message Event, Message Flow, and Sequence Flow

SupportFlow	If source is a material resource	Association
ExternalConnector		Participant (Pool) (Black Box)
ProcessConnector		Call Activity
InternalConnector		Participant (Pool) (Black Box)

4.3.2.2 EA* to BPMN Collaboration Diagram – Transformation Rules

Atomic ExtendedActivity

- A Human resource is responsible for the realization of the ExtendedActivity. In this case the atomic ExtendedActivity is mapped to a UserTask.
- An IT resource is responsible for the realization of the ExtendedActivity. In this case the atomic ExtendedActivity is mapped to a ServiceTask.

ControlFlow

- Source is an ExternalConnector or InternalConnector and target is an "atomic" ExtendedActivity. In this case it is mapped to a MessageFlow.
- Source is an ExternalConnector or InternalConnector and target is a "structural" ExtendedActivity. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.
- Source is a ProcessConnector or ExtendedActivity. It is a "1 to n relation", in which the "Control" Flow is mapped to a combination of DataObject and two Associations.

OutputInputFlow

- Source is an ExternalConnector or InternalConnector and target is an atomic ExtendedActivity. In this case the "OutputInput" Flow is mapped to a MessageFlow**.**
- Source and target are of type "atomic" ExtendedActivity and don't belong to the same organization. In this case it is mapped to a MessageFlow.
- Source is an atomic ExtendedActivity and target is an External or InternalConnector. In this case it is mapped to a MessageFlow.
- Source is a ProcessConnector, ExtendedActivity, or LogicalOperator, target is also one of these three options, and both source and target belong to the same organization. In this case it is mapped to SequenceFlow.
- Source is an ExtendedActivity, ProcessConnector, or LogicalOperator, target is also one of these three options, and source and target don't belong to the same organization. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow, throwing MessageEvent, and a SequenceFlow.
- Source is a structural ExtendedActivity or LogicalOperator, and target is an ExternalConnector or InternalConnector. This case is a "1 to n" relation, in which the "Control" Flow is mapped to a combination of MessageFlow, throwing MessageEvent, and a SequenceFlow.

- Source is an ExternalConnector or InternalConnector and target is a structural ExtendedActivity or LogicalOperator. This case is a “1 to n” relation, in which the “Control” Flow is mapped to a combination of MessageFlow, catching MessageEvent, and a SequenceFlow.

SupportFlow

- Source is a Material resource. In this case it is mapped to an Association**.**

4.4 Example

The concept of electronic marketplace (e-marketplace) has been based on grouping new and/or used products coming from several sellers on a unique internet platform and under the same e-commerce catalog. It needs to be assumed that it is solely a trading platform (it is an intermediate), the e-market itself does not sell nor buy directly physical products or services traded on the platform. On one hand e-marketplaces and supplier directories are B2B Internet platforms. Several enterprises took this opportunity to extend their offer. In France, major e-selling platforms are moving progressively to integrate this service offer such as CDiscount, Rue du Commerce, La Redoute, Brandalley, and PixMania who have initiated this e-commerce configuration since 2010. This selling concept fits the service orientation. It proposes two service interfaces, one dedicated to sellers and one to clients. A solution of export product catalog is embarked on marketplaces for a win/win strategy. The seller can reuse its catalog and be part of several marketplaces.

The example introduced in this section presents one of the processes value chains realized within an e-marketplace collaborative network. It details the process of purchasing products using a marketplace website. The marketplace is maintained by a broker agent that offers services for customers who choose, configure, and buy their products online. On the other hand, sellers are targeting customers and selling their products via the broker. As a result a collaborative network, formed of the broker website, sellers, and delivery companies, is offering a service to online customers. The business model is assisted in its transformation for generating the service platform that will be implemented. The following is an example of a private sale e-marketplace purchase process model which was modeled and transformed by using the SLMToolBox. It formalizes the business considerations captured with the EA* language (figure 41), then, in order to prepare the definition of the electronic platform, the SLMToolbox has transformed the model to BPMN 2.0 diagram (figure 42) according to the rules described in section 3.4.3.1.1. In details, in this model, the customer logs into with his user account, browses available brands, chooses a brand, browses brand’s available product, and configures his product (color, size, etc.). When the customer terminates the configuration, the broker agent verifies product's availability and delivery details from the seller company. These details will be transferred to the customer, who will decide to validate his basket or not. Then he can either pay for his products or go back to choose other products. The figures show that the SLMToolBox transformation has been able to identify the partners and to isolate the services they solicit or generate. In the BPMN model (Figure 42) the lanes on the upper side represent the B2C link and service required. On the lower side the lanes represent the collaborative network within a B2B relation where a competition is done between sellers to provide better proposition to client demand (e.g. about the delivery time and price). The goal was to detail the service system to set up between partners and identify the service product to be exchanged between them. The sequence of action is clearly defined in an unambiguous model. In particular, the data type to be exchanged is identified and the synchronization of the partners in the process flow can be used in order to orchestrate the services process between partners in the service system. Nevertheless, the transformation from EA* diagram to BPMN

diagram, is a passage from the bottom BSM level to top TIM level and thus to a different and a more specific level of details. As a result, the obtained BPMN diagram should be enriched at the top TIM level to satisfy the requirements of this level. The new structure is missing some technical information that will come from technical constraint of the level even if the tool prepares the objects to handle these concepts some semantic enrichment cannot be done automatically.

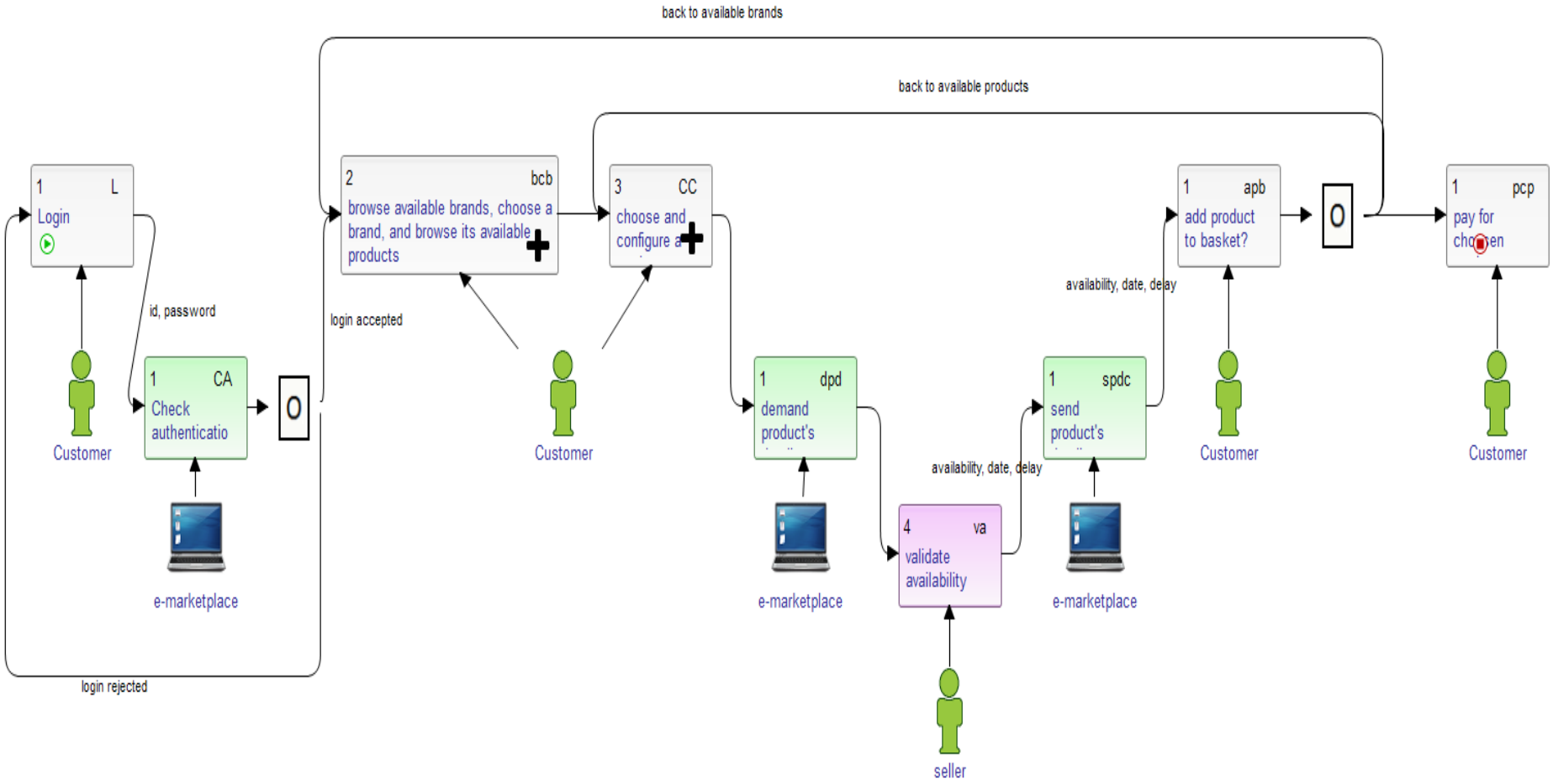


Figure 41 EA* e-marketplace purchase process

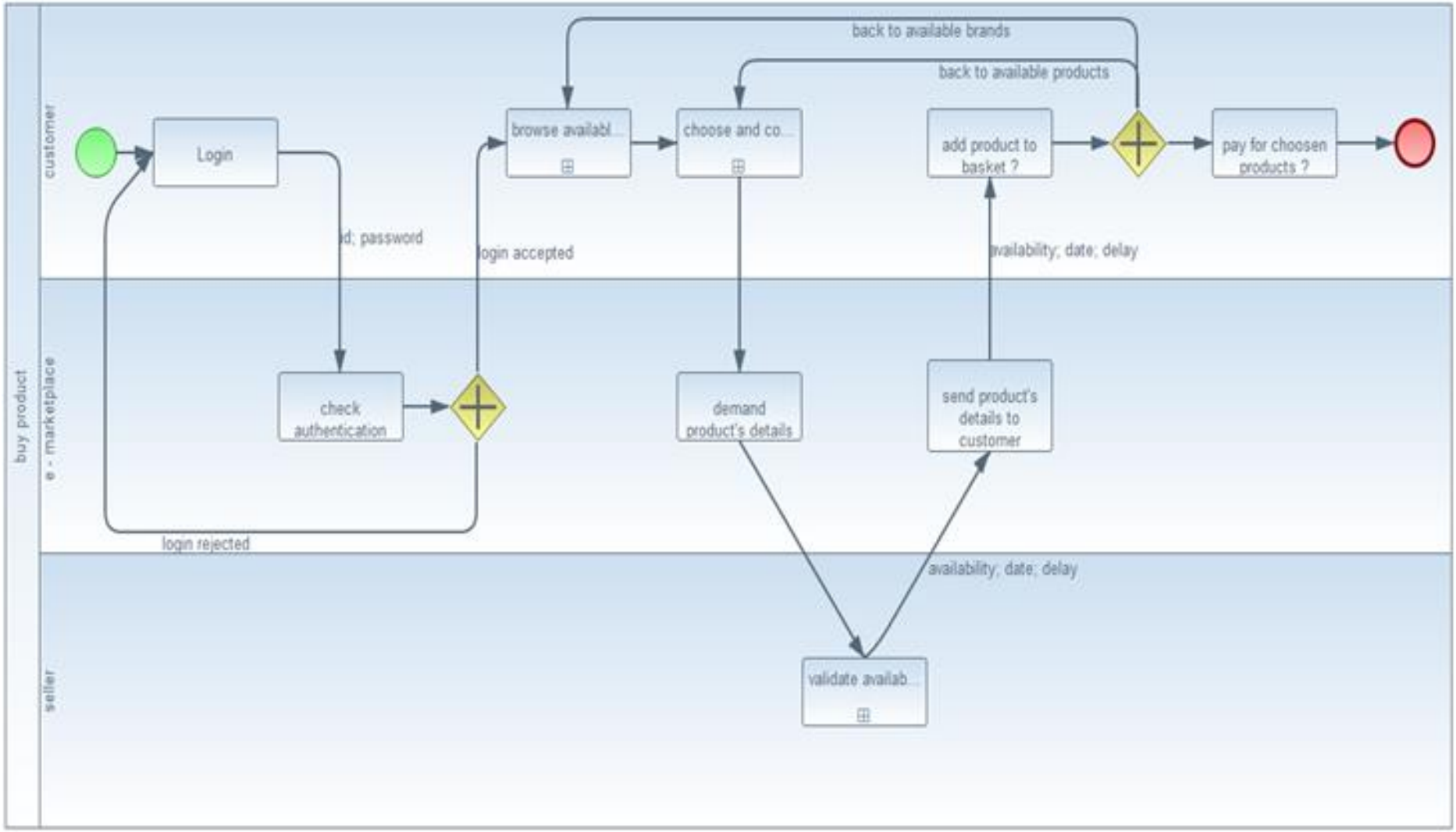


Figure 42 BPMN e-marketplace purchase process

5. Conclusion

In this chapter we presented the Model Driven Service Engineering Architecture (**MDSEA**) which is targeted to the representation of service systems and the management of certain aspects in the service's lifecycle. We detailed the three abstraction levels of MDSEA and the proposed modeling languages at each level. Besides, we introduced the Extended Actigram Star (**EA***) process modeling language, that we specified and developed based on the GRAI Extended Actigram language. The scope and overview of the language were presented in addition to the languages abstract and graphical syntaxes. The last section of the chapter consisted of a **model transformation** from EA* to BPMN in the frame of MDSEA. The transformation architecture and mappings were introduced with a case study example.

While this chapter was concerned with service modeling and model transformation, the next chapter will target service engineering, and in particular simulation in the frame of MDSEA.

Simulation and Model transformation from BPMN to DEVS

This chapter will illustrate service engineering in the frame of MDSEA, and in particular it will present the simulation of service systems based on the DEVS formalism and model transformation..

1. Introduction

Industrial enterprises have gradually moved their goals towards production of physical products supplemented by intangible services to differentiate themselves in a compatible market. The study of these services, their set up, and the evaluation of their efficiency is a rising research domain. To remain competitive, a company must differentiate itself from other competitors. Since improving the product's performance can reach some limits, one open solution is to improve the enterprise service system and redefine its business processes. Simulation of business processes answers this issue through analyzing these processes and concluding of they meet the desired objectives.

Simulation is the imitation of the operation of a real-world process or system over time [Banks et al, 2006]. The act of simulation necessitates representing specific characteristics or behaviors of a system. Simulation is used in several contexts (natural, human, technological, business, etc…). It offers a complete or partial study of the system in question, representing its characteristics, behaviors, interactions, communication with external environment. Scientists and engineers have long used models to better understand the system they study, for analysis and quantification, performance prediction and design. Real-world's systems are translated into models as virtual systems in order to conduct virtual experiments (simulation).

Depending on the purpose of simulation, in some cases performance indicators are defined for a system in order to deduce its progress to achieve its goals. In the world of manufacturing and business, simulation is starting to gain an increasing role and attention for its major role in decision making, risk studies, performance analysis and business model validation. Key performance indicators (KPIs) help an organization define and measure progress toward organizational goals. Once an organization has analyzed its mission, identified all its stakeholders, and defined its goals, it needs a way to measure progress toward those goals. KPIs are typically used for that purpose as measurements that are quantifiable, agreed to beforehand, and reflect the critical success factors of an organization. They will differ depending on the organization. From this perspective, a business model can be simulated while measuring the identified KPIs.

Once a service is designed or modeled, its implementation may be capable of delivering the desired output but not for the expected cost or within the desired timeframe. Also, a service that has been designed may be functional but may not be optimal. The analysis and understanding at the design stage of the service help in its optimization. Simulation is the process of virtualizing real world models in order to test their correctness, effectiveness and efficiency in response to specific problem space. It is composed of experiments in order to determine how the system can be improved, evolved, and to interpret how future changes will affect the modeled system. After problem definition, abstraction and modelling, the model can then be run and tested to assess its behavior in particular circumstances, i.e. when particular objects and entities in the model are given particular values. Results are then interpreted, analyzed, and then decisions to reach model and system optimization. A simulation model should incorporate the performance dimensions one is interested in. In most cases it should be possible to simulate time and cost aspects. Other relevant performance dimensions are quality and flexibility.

The questions that researchers and engineers have tried to answer were not targeted on the benefits and positive outcomes of simulation, but rather it focused more on "what kind of systems to simulate?", "how to conduct simulation?", and "what frameworks and tools to use?" This section is concerned with business process simulation in the frame of MDSEA. It defends and introduces the use of DEVS formalism and presents **how simulation is conducted in MDSEA**.

2. Problem

Model Driven Service Engineering Architecture (MDSEA) defines three abstraction levels (BSM, TIM and TSM) and specifies the modelling languages to use for modelling a service system. The defined modelling languages lead to a distinction between static and dynamic service system modelling. Static model is more structural than behavioral, helps in depicting static constituents of the system, rigid as it is time independent view of a system, and can't be changed in real time. On the other hand, a Dynamic model is a representation of the behavior of the static components of the system, and consists of a sequence of operations, state changes, activities, and interactions. Dynamic model is flexible as it can change with time as it shows what an object does with many possibilities that might arise in time. Business Process Modelling (BPM) [Cardoso et al. 2013] results in a representation of an organization's business processes to be analyzed and improved [Weske 2007]. Business process's models provide a suitable dynamic view, but frequently missing the temporal dimension to express output performance such as an expected cost or a desired duration. In detail, the impact of correct or incorrect behavior of complex models over time is not clearly visible using static view. This issue can be solved by running a business process simulation for analyzing and understanding the business process model according to its dynamic. In MDSEA, two modelling languages are used for business process modelling: Extended Actigram Star at BSM level ant BPMN at TIM level. At BSM level, the level of information is general and more addressed to business models and details. These business models developed at the BSM level lack some detailed specifications of the structure and functionality of the service system important for running accurate simulations. As a result, BPMN models developed at TIM levels are chosen to be the subject of simulation to study the behavior of the service system being developed. BPMN models developed in the SLMToolBox need to be transformed to a simulation models in order to be simulated and analyzed.

In [Zacharewicz et al, 2008] an automatic transformation of a Workflow into a G-DEVS model has been defined. In the context of BPMN to DEVS transformation, authors in [Cetinkaya et al, 2012] and [Mittal et al, 2012] presented a Model Driven Development (MDD) framework for modelling and simulation (MDD4MS). In this framework they defined a model to model transformation from BPMN as a conceptual modelling language to DEVS as a simulation model specification. BPMN and DEVS Meta-models were presented. In addition, a set of transformation rules were defined in order to transform BPMN models into DEVS models. According to these rules, some BPMN concepts (Pool, Lane, SubProcess) were mapped to DEVS coupled component, while Task, Event (Start, End, and Intermediate), and Gateway were mapped to DEVS atomic component.

Comparing the BPMN metamodel defined with the latest version of BPMN 2.0 metamodel [OMG-2, 2011] we can conclude that several concepts are missing and thus were not transformed into their corresponding DEVS concept. Authors didn't mention the different types of BPMN Tasks (User Task, Manual Task, Service Task…) and BPMN Intermediate Events (Message, Signal…) that can be mapped differently when transformed into DEVS

concepts. The difference would be in the number of states forming each DEVS Atomic Model. Based on these remarks, the work presented in this paper takes into consideration these points in an attempt to benefit from previous work and propose new mapping and transformation rules.

3. DEVS

DEVS [Zeigler, 2000] (presented earlier in section 2.4.3) is the most general formalism for discrete event system modelling. It allows representing any system provided that it performs a finite number of changes in finite intervals of time. Thus, not only Petri–Nets, State–charts, Event–Graphs and other discrete event languages but also all discrete time systems can be seen as particular cases of DEVS [Zeigler, 1993].

3.1 Basic DEVS characteristics

The DEVS simulation is a mathematical paradigm with well-defined concepts of coupling of components, hierarchical, modular model construction, support for discrete event approximation of continuous systems and an object-oriented substrate supporting repository reuse. DEVS is characterized by the following:

- The notion of time is well recognized in DEVS. Time is tracked through the representation of the system (states and atomic models) and the running of a simulation where simulators clocks are always updated in order to keep track of the elapsed and actual times.
- Hierarchical compositions of models in order to define composite models. DEVS possess Coupled and Atomic Models, where a coupled model is composed of DEVS Models (Atomic or Coupled). This hierarchical composition is essential for a component view modeling of a system (system is recognized as a group of components)
- Separation between Model and Simulation concepts. Where a descriptive model of the system's behavior is developed independent of Simulation concepts (Coordinators and Simulators).

3.2 Simulation of DEVS Model

One of the most important features of DEVS is that very complex models can be simulated in a very easy and efficient way. The basic idea for the simulation of a coupled DEVS model can be described by the following steps:

- Look for the atomic model that, according to its time advance and elapsed time, is the next to perform an internal transition. Call it d* and let tn be the time of the mentioned transition.
- Advance the simulation time t to t = tn and execute the internal transition function of d*.
- Propagate the output event produced by d* to all the atomic models connected to it executing the corresponding external transition functions. Then, go back to step 1.

One of the simplest ways to implement these steps is writing a program with a hierarchical structure equivalent to the hierarchical structure of the model to be simulated. A routine called *DEVSsimulator* is associated to each *atomic DEVS model* and a different routine called

DEVS-coordinator is related to each *coupled DEVS model*. At the top of the hierarchy there is a routine called *DEVS-root-coordinator* which manages the global simulation time. The simulators and coordinators of consecutive layers communicate with each other with messages. The coordinators send messages to their children so they execute the transition functions. When a simulator executes a transition, it calculates its next state and –when the transition is internal– it sends the output value to its parent coordinator. In all the cases, the simulator state will coincide with its associated atomic DEVS model state. The figure below shows a hierarchical model (Coupled Model) being associated to a Coordinator and it reveals how atomic Models are associated to (atomic) Simulators.

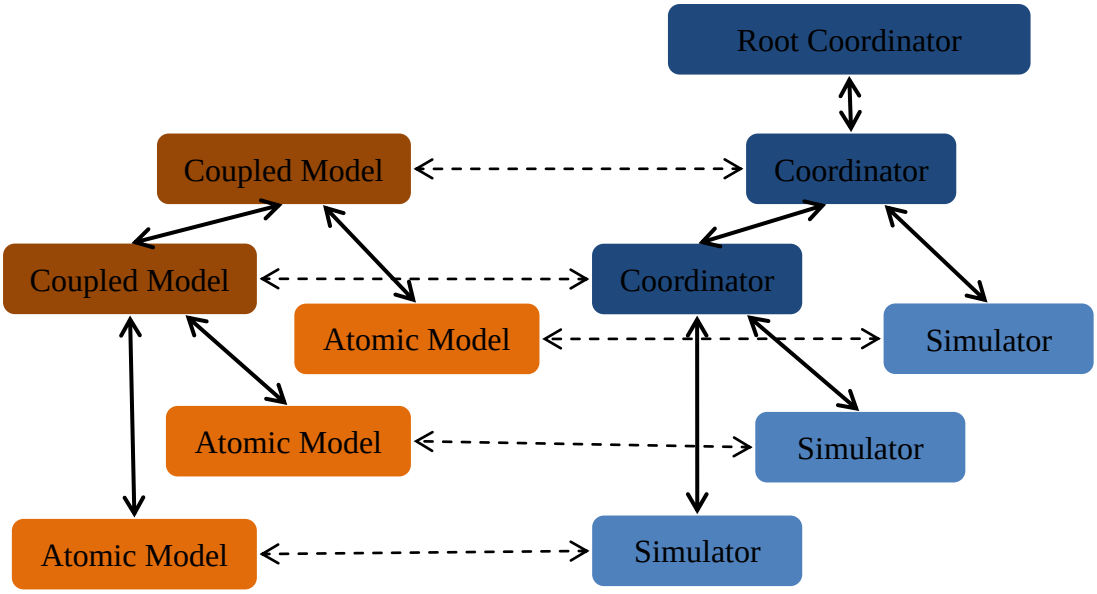


Figure 43 Relations Simulator-Model (a)

When a coordinator executes a transition, it sends messages to some of its children so they execute their corresponding transition functions. When an output event produced by one of its children has to be propagated outside the coupled model, the coordinator sends a message to its own parent coordinator carrying the output value. Each simulator or coordinator has a local variable *tn* which indicates the time when its next internal transition will occur. In the simulators, that variable is calculated using the time advance function of the corresponding atomic model. In the coordinators, it is calculated as the minimum *tn* of their children. Thus, the tn of the coordinator at the top is the time at which the next event of the entire system will occur. Then, the root coordinator only looks at this time, advances the global time t to this value and then it sends a message to its child so it performs the next transition, and then it repeats this cycle until the end of the simulation.

4. Transformation BPMN to DEVS

The DEVS-based simulation of BPMN models requires a transformation process. The modelling elements of BPMN have to be mapped to DEVS components in order to be able to simulate their behavior in a DEVS simulation environment. The transformation from BPMN to DEVS is based on the metamodel approach (the same approach used for the transformation from EA* to BPMN). For the mapping to be established and created it is important to well define the source and target metamodels involved in this transformation. The BPMN 2.0 source metamodel is defined in [OMG-2 2011]. There is no standard metamodel for DEVS,

all metamodels are the result of research works done in universities and research group without being standardized. From these efforts we can distinguish the work done in [Garredu et al, 2012] which defined a DEVS metamodel based on Model Driven Engineering specifications. The following section presents a DEVS metamodel used in the frame of the transformation from BPMN to DEVS, and built from the understanding of the DEVS formalism.

4.1 DEVS Metamodel

As seen in the section Classic DEVS Formalism, DEVS is formed basically from models (atomic and coupled), ports (input and output), couplings (external-input, external-output, and internal), transitions (internal and external), and states. there are two types of models: atomic and coupled models. Both types have input and output ports which define entry and exit points of the model and stores the values received/sent by/from models. Each model has a list of Input Ports and Output Ports. If we analyze the atomic model definition from previous section we can distinguish four main methods: internal transition, external transition, output, and time advance. In addition to these methods, atomic model is characterized by the possession of states. These states are connected via transitions which can be internal or external transitions. As for the coupled model, it is a decomposition of DEVS models (atomic or coupled) and Couplings connecting output and input ports. Three types of coupling between ports: External Input Coupling (connection between input port of the coupled model and an internal component), External Output Coupling (connection between internal components and the output port of the coupled model), and Internal Coupling (connection between internal components). Figure 44 is a simplified DEVS Metamodel proposed which presents the basic classes used in the transformation.

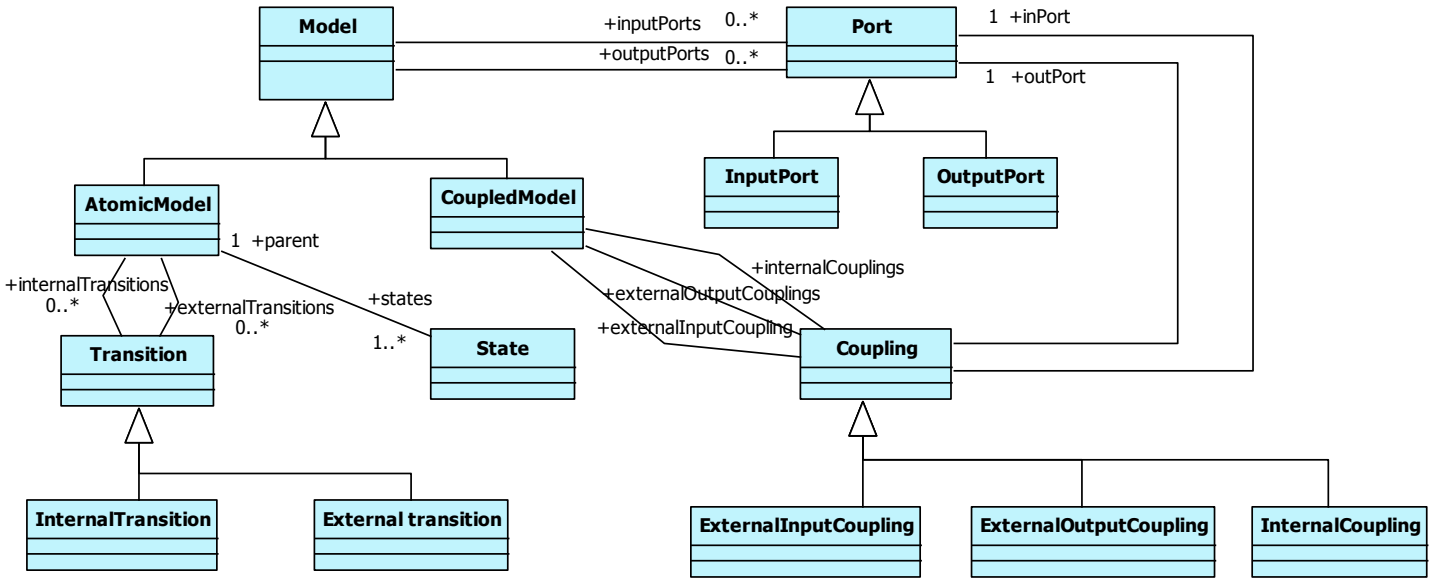


Figure 44 DEVS simplified Metamodel

4.2 Transformation rules

The role of mapping in model transformation is to define links between concepts and relations from both metamodels (BPMN and DEVS). In [Deniz et al. 2012], a first mapping was proposed by the authors. Nevertheless, this early mapping didn't distinguish all the various types of tasks and events existing in BPMN 2.0 which differ with respect to the potential situations a task might treat. To complete this approach, different types of tasks are detailed

(Receive task, Send Task, User Task, Service Task, and Manual Task); all of these tasks are mapped to "DEVS Atomic Model" concept but with different local behavior. This is also applied to intermediate events (Receiving and Sending Messages). In the following sections, the mapping of concepts will be elaborated using graphical notations extracted from the SLMToolBox editors.

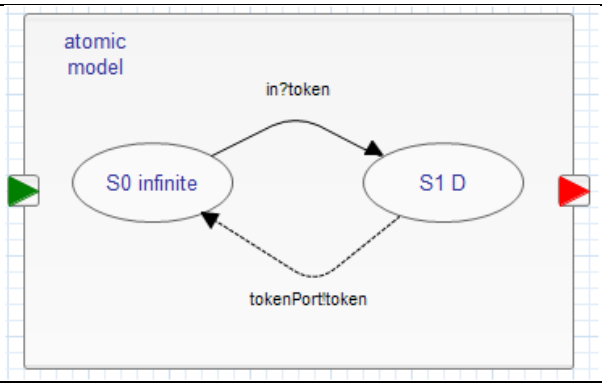
4.2.1 BPMN Task to DEVS Atomic Model

A BPMN Task is an atomic Activity within a process flow. A task is used when the work in the process cannot be broken down to finer levels of details. Different types of tasks are identified to separate the types of inherited behavior that tasks might represent.

BPMN Task

A Task is transformed into DEVS Atomic model possessing two states (the initial state S0 which is a passive state with an infinite *sigma* and a state S1 with *sigma* equals to D time unit). These two states are connected with internal and external transitions.

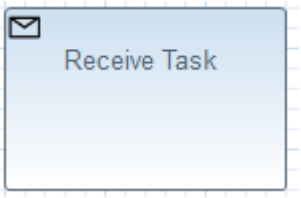
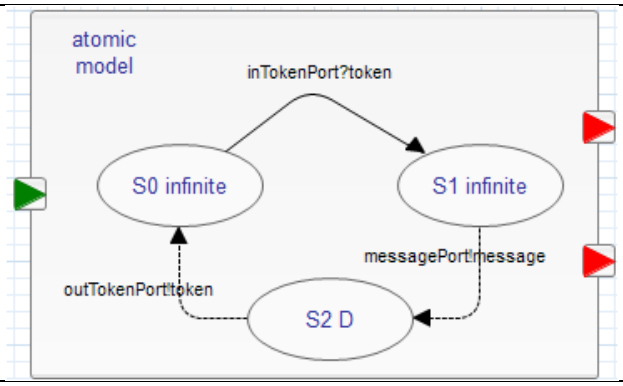
Table 34 BPMN Task to DEVS Atomic Model

BPMN Task	DEVS Atomic Model
	

BPMN Receive Task

A Receive Task is a simple task that is designed to wait for a message to arrive from an external participant. Once the task has been received, the Task is completed. The Receive Task is transformed to a DEVS Atomic Model possessing three states (an initial passive state S0 with an infinite *sigma*, state S1 with also an infinite *sigma*, and a state S2 with *sigma* equals to D time unit). The DEVS Atomic model is initially at its unit state waiting for an external event. When a token arrives, the model changes its state from state unit to state S1 which in its tern will wait for another external event (message arrival). Upon receiving the message, the model will change state from S1 to state S2 with sigma equals to D (time unit). D represents the time needed for the execution of the atomic model after the reception of the message.

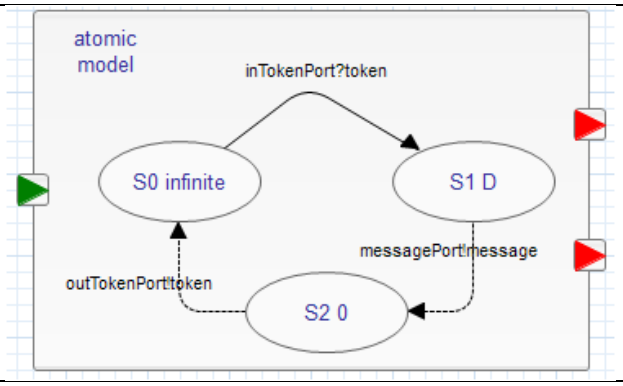
Table 35 BPMN Receive Task to DEVS Atomic Model

BPMN Receive Task	DEVS Atomic Model
	

BPMN Send Task

A Send Task is a simple task that is designated to send a Message to an external Participant (relative to the process). Once the message has been sent, the task is completed. A Send Task is transformed to DEVS Atomic Model possessing three states (an initial passive state S0 with infinite sigma, a state S1 with sigma equals to D time unit, and a state S2 with sigma equal to 0). The DEVS Atomic model is initially at its inti state waiting for a token arrival. After the token has arrived, the model changes state to state S1 which will take D (time unit) before the model sends the message and changes its state to S2. The model will not wait since ta(S2) = 0 and will send a token before moving back to its initial state.

Table 36 BPMN Send Task to DEVS Atomic Model

BPMN Send Task	DEVS Atomic Model
	

4.2.2 BPMN Event to DEVS Atomic Model

An Event is something that “happens” during the course of a Process. These Events affect the flow of the Process and usually have a cause or an impact and in general require or allow for a reaction.

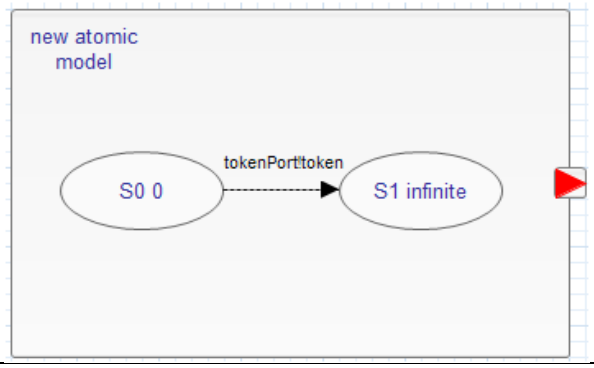
4.2.2.1 BPMN Start Event to DEVS Atomic Model

The Start Event indicates where a particular Process will start. In terms of Sequence Flows, the Start Event starts the flow of the Process, and thus, will not have any incoming flows

BPMN None Start Event

None Start Event is a basic Start Event without any triggers. A Non Start Event is transformed into a DEVS Atomic Model with two states (an initial state S0 with *sigma* equal to 0 and a passive state S1 with an infinite *sigma*). The Atomic Model will send a token and then changes its state to state S1.

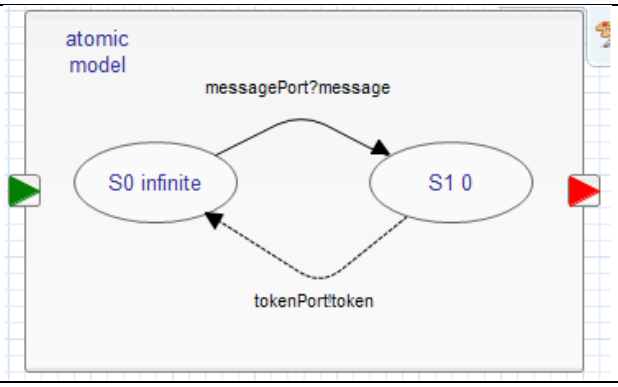
Table 37 BPMN Start Event to DEVS Atomic Model

BPMN None Start Event	DEVS Atomic Model
	

BPMN Message Start Event

A Message Start Event implies that A Message arrives from a Participant and triggers the start of the Process. The Message Start Event is transformed to a DEVS Atomic Model with two states (an initial passive state S0 with an infinite *sigma* and a state S1 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of message. When the message arrives it changes its state to state S1, and since the waiting time is 0 it send a token via the token Port and moves again to its passive state S0.

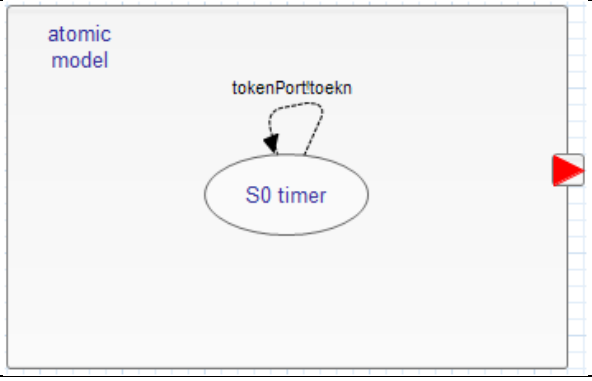
Table 38 BPMN Message Start Event to DEVS Atomic Model

BPMN Message Start Event	DEVS Atomic Model
	

BPMN Timer Start Event

A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the start of the Process. The Message Start Event is transformed to a DEVS Atomic Model possessing a single state S0 with a sigma equals to timer (time unit). The DEVS Atomic Model will wait a time equals to timer before sending the token. After the token is sent it will wait again for a time equals to timer before sending the next token.

Table 39 BPMN Timer Start Event to DEVS Atomic Model

BPMN Timer Start Event	DEVS Atomic Model
	

4.2.2.2 BPMN Intermediate Event to DEVS Atomic Model

Intermediate Event indicates where something happens (an Event) somewhere between the start and end of a Process. It will affect the flow of the Process, but will not start or (directly) terminate the Process. There are twelve types of Intermediate Events in BPMN: *None*, Message, Timer, Escalation, Error, Cancel, Compensation, Conditional, Link, Signal, Multiple, and Parallel Multiple. There are two ways that Intermediate Events are used in BPMN:

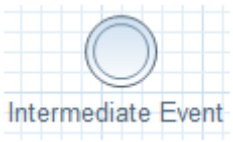
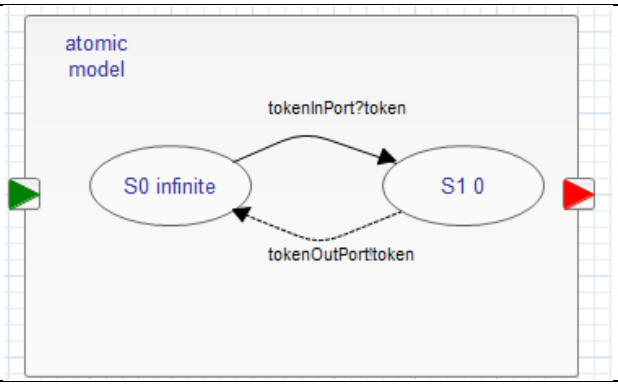
- An Intermediate Event that is placed within the *normal flow* of a Process can be used for one of two purposes. The Event can respond to ("catch") the Event *trigger* or the Event can be used to set off ("throw") the Event *trigger*.
- An Intermediate Event that is attached to the boundary of an Activity can be only used to "catch" the Event *trigger*.

This section covers two types of intermediate events: None and throw/catch Message.

BPMN None Intermediate Event

The None Intermediate Event is defined as throw event but with a non-defined trigger to be thrown (it is used for modelling methodologies that use events to indicate some change of state in the Process). The None Intermediate Event is transformed to a DEVS Atomic Model with two states (an initial passive state S0 with an infinite *sigma* and a state S1 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1, and since the waiting time is 0 it send a token via the token Port and moves again to its passive state S0.

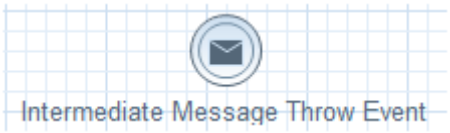
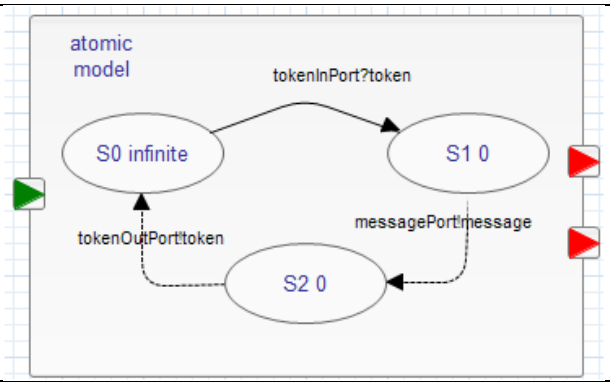
Table 40 BPMN None Intermediate Event to DEVS Atomic Model

BPMN None Intermediate Event	DEVS Atomic Model
	

BPMN Message Intermediate Throw Event

Message Intermediate Throw Event is used to throw a message. The Message Intermediate throw Event is transformed to a DEVS Atomic Model with three states (an initial passive state S0 with an infinite *sigma* and states S1and S2 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1, and since the waiting time is 0 it send a message via the message Port and moves to S2. Then the DEVS Model will send a token via the token port and changes its state to the initial state.

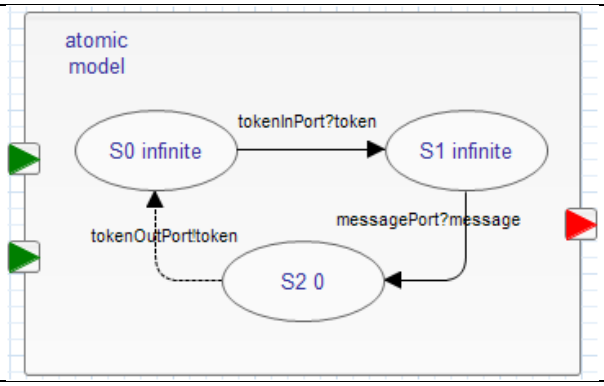
Table 41 BPMN Message Intermediate throw Event to DEVS Atomic Model

BPMN Message Intermediate Throw Event	DEVS Atomic Model
	

BPMN Message Intermediate Catch Event

Message Intermediate Catch Event is used to throw a message. The Message Intermediate Catch Event is transformed to a DEVS Atomic Model with three states (an initial passive state S0 with an infinite *sigma*, states S1with an infinite *sigma*, and S2 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1 and waits the arrival of a message. After the message arrival, the DEVS Atomic Model changes its state to S2. Then it will send a token via the token port and changes its state to the initial state.

Table 42 BPMN Message Intermediate catch Event to DEVS Atomic Model

BPMN Message Intermediate Catch Event	DEVS Atomic Model
	

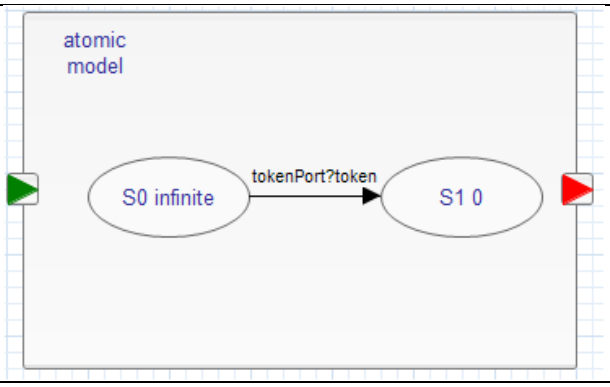
4.2.2.3 BPMN End Event to DEVS Atomic Model

End Event indicates where a Process will end. In terms of Sequence Flows, the End Event ends the flow of the Process, and thus, will not have any outgoing Sequence Flows—no Sequence Flow can connect from an End Event. There are different types of End Events that indicate different categories of results for the process. When a token arrives at an End Event, the result of the vent if any occurs and the token is consumed.

BPMN None End Event

The None End Event does not have a defined result. There is no specific Event Definition for None End Events. The End Event is transformed to a DEVS Atomic Model with two states (an initial passive state S0 with an infinite *sigma* and a state S1 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1.

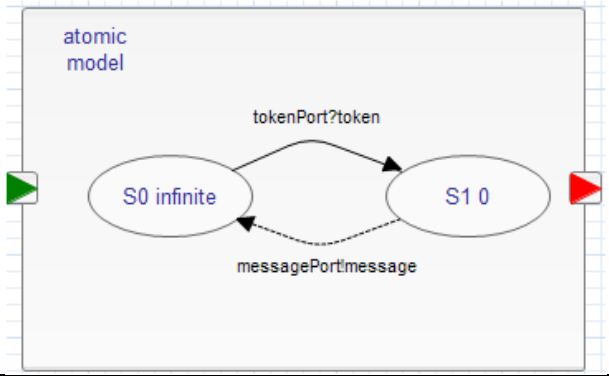
Table 43 BPMN End Event to DEVS Atomic Model

BPMN None End Event	DEVS Atomic Model
	

BPMN Message End Event

Message End Event indicates that a Message is sent to a Participant at the conclusion of the Process. The Message End Event is transformed to a DEVS Atomic Model with two states (an initial passive state S0 with an infinite *sigma* and a state S1 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1, and since the waiting time is 0 it send a message via the message Port and moves again to its passive state S0.

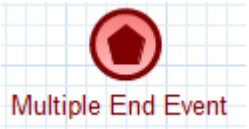
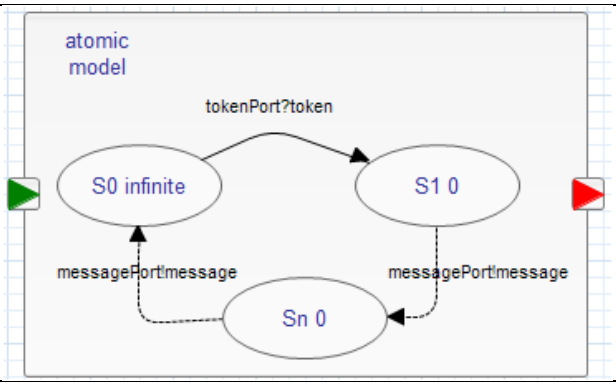
Table 44 BPMN Message End Event to DEVS Atomic Model

BPMN Message End Event	DEVS Atomic Model
	

BPMN Multiple End Event

Multiple End Event means that there are multiple consequences of ending the process. All of them will occur such as sending multiple messages. If an End Event has more than one associated Event Definition, then the event will be considered a Multiple End Event. The Multiple End Event is transformed to a DEVS Atomic Model with states depending on the event definitions associated to it (an initial passive state S0 with an infinite *sigma* and n states S1…Sn with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1, and since the waiting time is 0 it send a message via the message Port and moves to the next state (if any). At the state Sn a message is sent via the message Port and then it changes state to the initial passive state S0.

Table 45 BPMN Multiple End Event to DEVS Atomic Model

BPMN Multiple End Event	DEVS Atomic Model
	

4.2.3 BPMN Gateway to DEVS Atomic Model

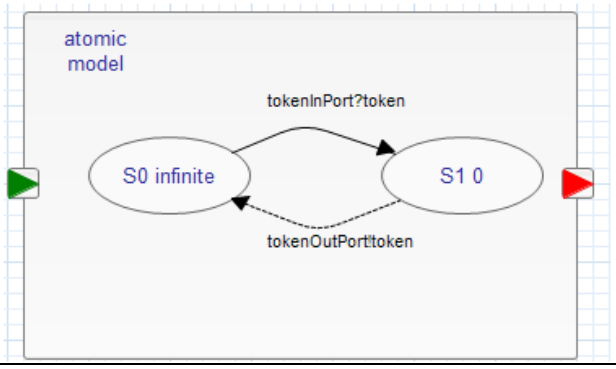
Gateways are used to control how the Process flows (how Tokens flow) through Sequence Flows as they converge and diverge within a Process. The term “gateway” implies that there is a gating mechanism that either allows or disallows passage through the Gateway--that is, as tokens arrive at a Gateway, they can be merged together on input and/or split apart on output as the Gateway mechanisms are invoked.

BPMN Diverging Exclusive Gateway

A diverging Exclusive Gateway (Decision) is used to create alternative paths within a Process flow. This is basically the “diversion point in the road” for a Process. For a given instance of the Process, only one of the paths can be taken. The Diverging Exclusive Gateway is transformed to a DEVS Atomic Model with two states (an initial passive state S0 with an

infinite *sigma* and state S1 with a *sigma* equals to 0). The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When the token arrives it changes its state to state S1, and since the waiting time is 0 it sends the token via the token Port and moves back to the initial state S0. Several DEVS Internal Couplings are connected to the tokenOutPort but only one coupling will continue the process (the one with a token).

Table 46 BPMN Exclusive Gateway to DEVS Atomic Model

BPMN Exclusive Gateway	DEVS Atomic Model
	

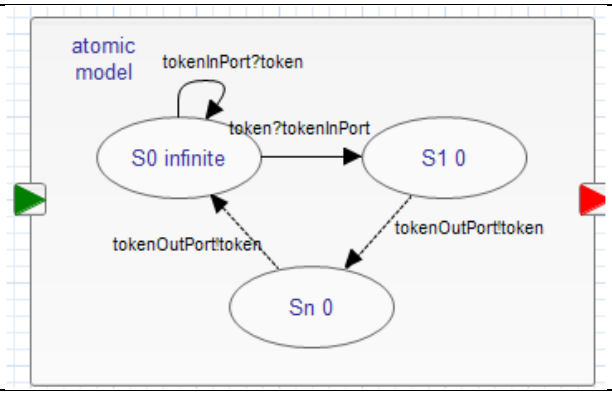
BPMN Converging Exclusive Gateway

Converging Exclusive Gateway is used to merge alternative paths. Each incoming Sequence Flow token is routed to the outgoing Sequence Flow without synchronization. The Converging Exclusive Gateway is transformed to the same DEVS Atomic Model as the Diverging one. The difference will be in the incoming/outcoming Internal Couplings connected to/from Input/Output Port.

BPMN Parallel Gateway

A Parallel Gateway creates parallel paths without checking any conditions; each outgoing Sequence Flow receives a token upon execution of this Gateway. For incoming flows, the Parallel Gateway will wait for all incoming flows before triggering the flow through its outgoing Sequence Flows. The Parallel Gateway is transformed to a DEVS Atomic Model with states depending on the outcoming flows associated to it (an initial passive state S0 with an infinite *sigma* and n states S1…Sn with a *sigma* equals to 0). The number of states n depends on the number of outgoing flows. The DEVS Atomic Model is first in its passive state waiting for the arrival of a token. When a token arrives and it is the last token to be received it changes its state to state S1else it will stay in its recent state S0 waiting the arrival of the last one. When the DEVS Atomic Model is in its state S1, it will send a token to the OutputPort and then changes it state to the next state. At the state Sn a token is sent via the OutputPort and then it changes state to the initial passive state S0.

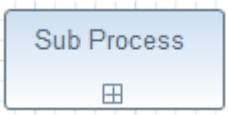
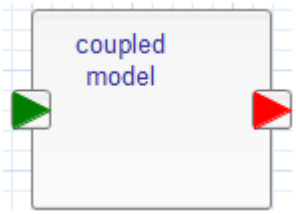
Table 47 BPMN Parallel Gateway to DEVS Atomic Model

BPMN Parallel Gateway	DEVS Atomic Model
	

4.2.4 BPMN Lane, Pool, and SubProcess to DEVS Coupled Model

Lanes, Pools, and SubProcesses are objects whose internal details can be modeled using other kinds of flowElements such as Activities, Gateways, and Events…These BPMN elements are transformed into DEVS Coupled Model.

Table 48 BPMN SubProcess to DEVS Coupled Model

BPMN SubProcess	DEVS Coupled Model
	

4.2.5 BPMN Flow to DEVS Coupling

Sequence Flows and Message Flows are used for connecting Flow Objects to each other. Sequence Flow is used to show the order of Flow elements in a Process, while Message Flow is used to show the flow of Messages between two Participants that are prepared to send and receive them. The following table presents the transformation of Sequence Flows and Message Flows to DEVS Couplings with the conditions that control such transformation.

Table 49 BPMN Flow to DEVS

BPMN Flow	Condition	DEVS
SequenceFlow	If source and target belong to the same lane	InternalCoupling
	If source and target belong to different Lanes or different Pools.	ExternalOutputCoupling, ExternalInputCoupling, InternalCoupling
MessageFlow		ExternalOutputCoupling, ExternalInputCoupling, InternalCoupling

5. DEVS Simulation

As previously described, the Discrete EVent System Specification (DEVS) is a mathematical

formalism for describing discrete event systems. The hierarchical and modular structure of a DEVS model is reflected in the classical specification of the DEVS simulators [Zeigler et al., 2000]. Each atomic model is associated with a simulator object. The simulator is controlled by sending messages such as "compute next state" and "compute next output", and it makes requests such as "get time of next event". A coordinator object is associated with each network model, and the coordinator can respond to the same types of messages as the simulator objects. The coordinator, as its name suggests, coordinates the execution of its coordinator(s) and simulator(s). The Root Coordinator is responsible for the execution of the simulation and it keeps track of results.

Figure 45 is a class diagram of DEVS Models and Simulators. It presents basic methods contained in Simulators and Models.

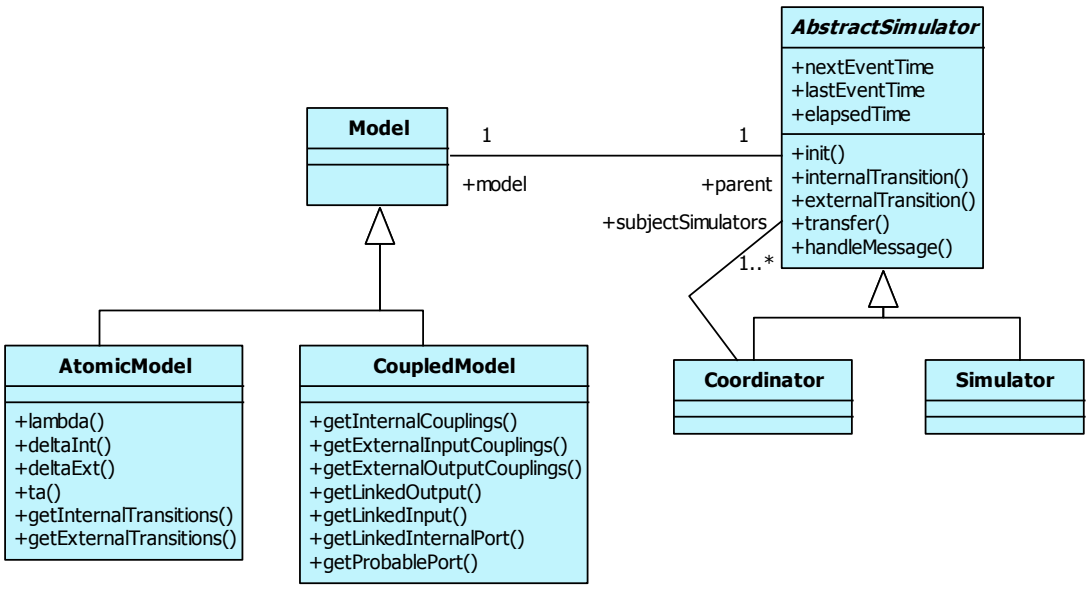


Figure 43 Relation Simulator-Model (b)

5.1 Execution

Execution of DEVS simulations is based on a specific protocol that orchestrates the execution of events. The protocol or scenario is based in the behavior of the root coordinator, coordinators, and simulators. This section presents the execution scenario which is implemented in the SLMToolBox (DEVS editor).

Root Coordinators and Simulators

The root coordinator is responsible for initializing the simulators' clock and running the simulation. The synchronization between different simulators is managed through the usage of messages which play an essential role in the communication between these simulators. Different types of messages are available for this purpose: IMessage (initialization message), SMessage (internal transition message), XMessage (external transition message), and YMessage (output message). Each message contains a time t and/or a Port p.

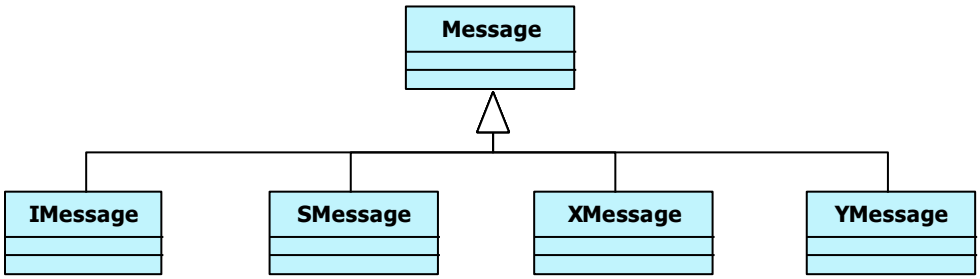


Figure 44 DEVS Message

Simulation starts by initializing the simulation clock of all simulators:

- Root Coordinator sends an IMessage to its Coordinator to initialize the clock of all simulators.
- Coordinator receives the IMessage sent by the Root Coordinator or its parent Coordinator; it sends the message to all of its child simulators and coordinators. Then it updates its *lastEventTime* (time of last event) to t and its *nextEventTime* (time of next event) to the Minimum nextEventTime of its children simulators.
- Simulator receives the IMessage sent by its parent Coordinator, it sets its *lastEventTime* to t and the *nextEventTime* to $t + ta(s)$ where s is its initial state.

Then an internal transition message (SMessage) is sent by the Root Coordinator:

- Root Coordinator sends a SMessage with time t to its Coordinator.
- Coordinator receives the SMessage sent by the Root Coordinator or its parent Coordinator, it then handles this message to a child simulator or coordinator whose *nextEventTime* = *t*. Then it updates its *nextEventTime* (time of next event) to the Minimum *nextEventTime* of its children simulators.
- Simulator receives the SMessage sent by its parent Coordinator; it sets its lastEventTime to t and asks its atomic model to execute the *lambda* and *deltaInt* functions. Then the simulator sets its *nextEventTime* to $t + ta(s)$ where s is its active state.
- Simulator signals his parent coordinator to handle a new YMessage which contains information about the port holding the data to deliver and the *lastEventTime*.
- Upon receiving the YMessage, the parent Coordinator searches the target port *tp* associated to the source port *sp* (contained in the YMessage) and asks the Abstract Simulator (Coordinator or Simulator) associated to the Model (Atomic or Coupled) containing the port *tp* to handle an XMessage.
- If the abstract simulator is a Coordinator, it searches the port linked (via an ExternalInputCoupling) to the port *tp* (included in the XMessage). Then it asks the Abstract Simulator (Coordinator or Simulator) responsible for the port to handle an XMessage.
- Else if the abstract simulator is a Simulator, it will ask the associated atomic model to execute the *deltaExt* function and set its last event to t and the *nextEventTime* to $t + ta(s)$ where *s* is its initial state.

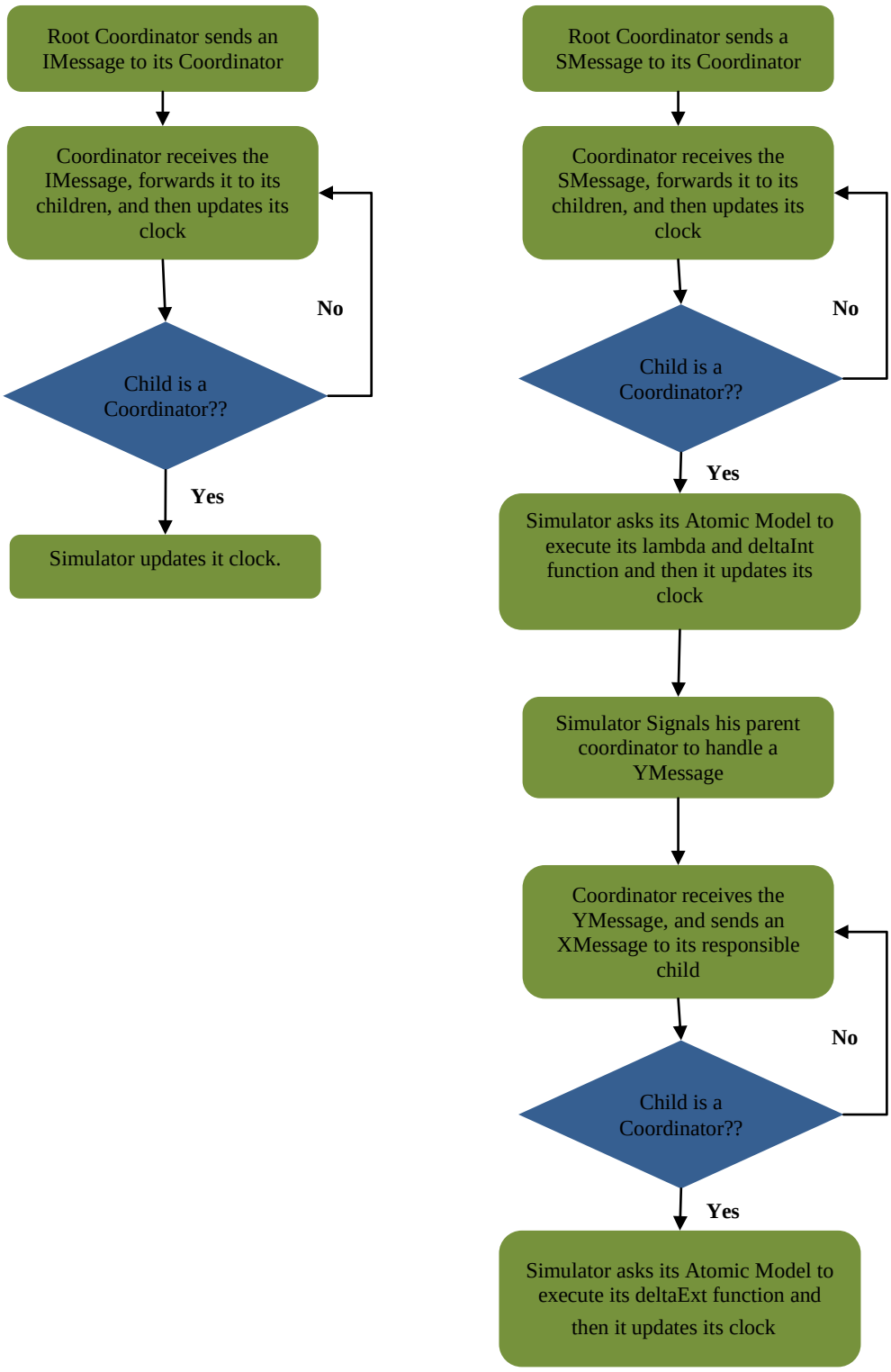


Figure 45 Simulation Algorithm

5.2 Simulation's profile and results

Business process simulation is based on several criteria or **performance indicators** which should be identified before running any simulation. These indicators represent inputs to simulation models which would be processed by the simulation engine. The performance indicators implemented, studied and used in the developed DEVS Simulator are **time** and **cost**. As a result, in order to simulate a process, the user is supposed to **manually enrich** the

DEVS model by time and cost estimations. Several values are needed before executing the simulation otherwise the simulation will not proceed:

- Estimated **cost** for the execution of every DEVS Model.
- Estimated **processing time** for every State presented in the DEVS diagram.
- **Probabilities** of divergent internal couplings.

Later the user is responsible for the definition of a simulation profile which is regarded to be essential for results analysis and interpretations. The **Simulation profile** is targeted to collect user's objectives just before running the simulation. These objectives depend on the performance indicators to be analyzed throughout the simulation. In our DEVS Simulator, the simulation profile consists of the number of instances to run and the optimal cost and time estimated for the process. An instance corresponds to a full execution of the process starting from its start event (entry point) and ending with its end event (exit point). The number of instances will permit the process to take different paths in case of divergence and thus analyze the costs and time needed of different paths. In addition, optimal costs and processing time (to be defined by the user) represent the **objectives** of the user or the waited results from the execution of the business process. Time and cost indicators will be interpreted by simulators throughout the simulation process and the results will be delivered at the end of the simulation to the user in form of a report.

Calculating time, cost, and Probabilities

Users estimate the cost of executing an Atomic Model, the processing time of states, and probabilities of divergent couplings. Calculating the execution time and cost of the process (from its entry to exit) is managed by the Root Coordinator. The root Coordinator runs several consecutive instances corresponding to the number of instances defined by the user in the simulation profile. Each instance is a complete execution which terminates by the process reaching an end. The Root Coordinator keeps traces (in an xml output file) of executed States and Models during the execution and organizes them in paths where every path corresponds to an execution of an instance. Processing time and costs of each path are also stored in every path. Figure 49 presents the algorithm implemented for running instances and generating the xml output file. The generated xml output file is used to generate a pdf report that contains graphs and simulation results.

Users define probabilities associated to Internal Couplings. These probabilities are relevant in case of divergent Internal Couplings. Figure 48 is an extraction of a DEVS diagram with two divergent Internal Couplings with probabilities 30% and 70%. The path to be taken during the execution is determined depending on the probability associated to each Internal Coupling.

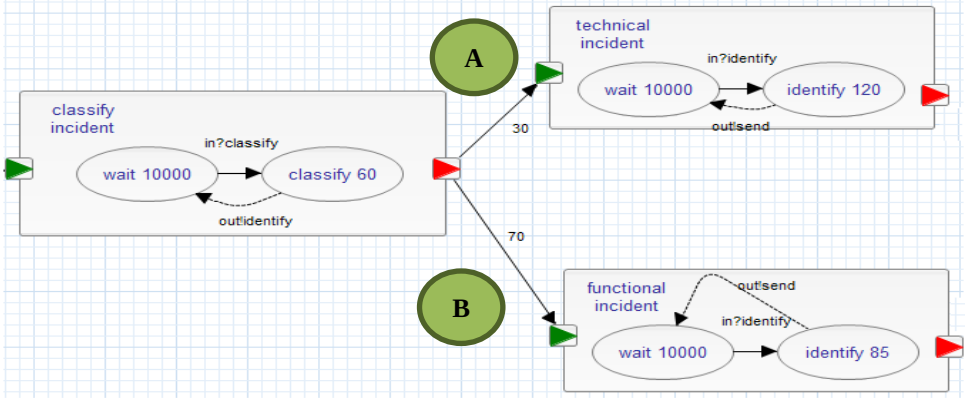


Figure 46 Calculating probabilities

In Figure 48 there are two probabilities **0.3** and **0.7**. A random number between 0.0 and 1.0 is generated using the Java *Math* library and its method ***Math.random()***. If the random number is *less* than 0.3 then path **A** is selected, else path **B**.

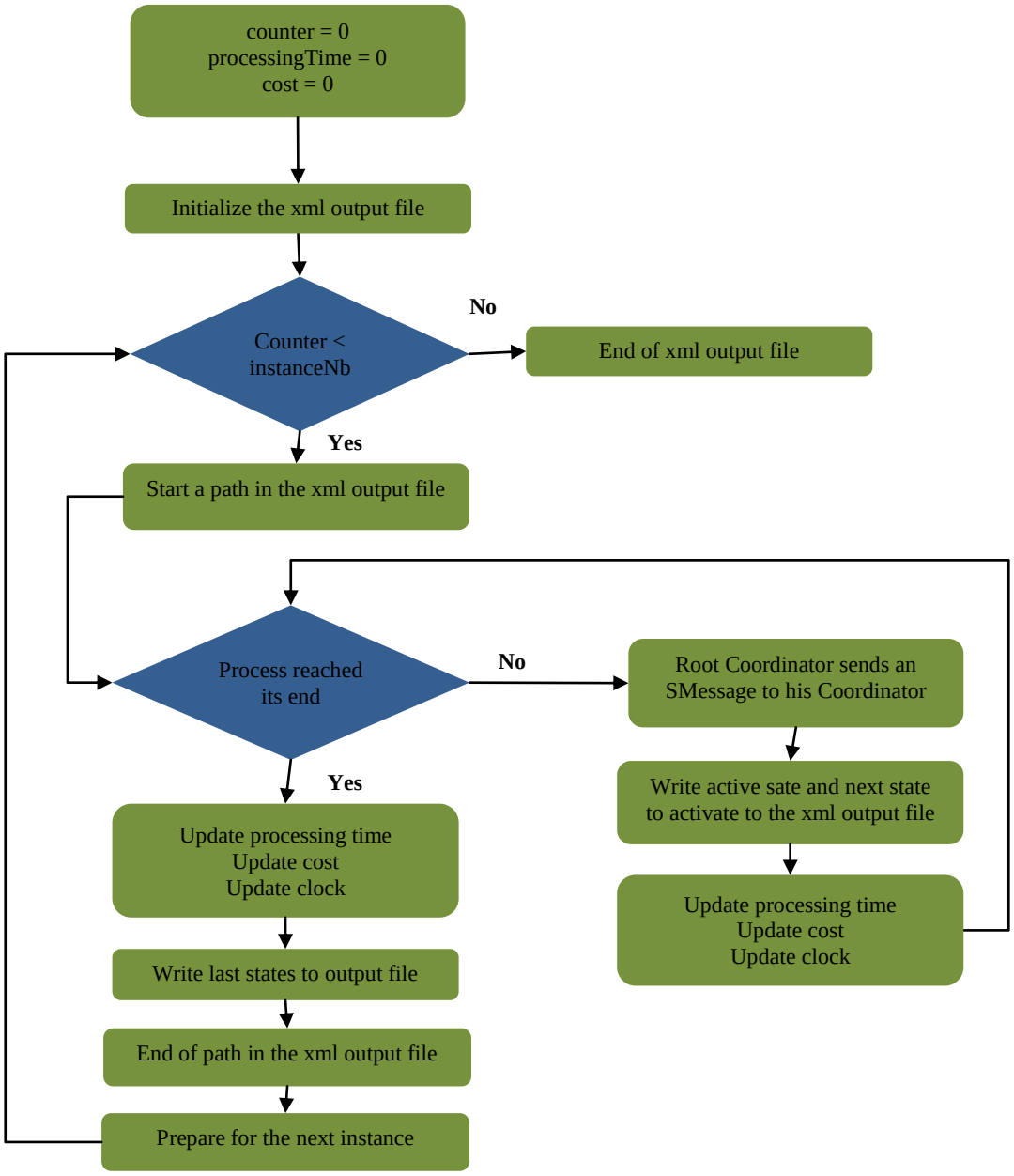


Figure 47 Calculating time and cost

5.3 Animation

Animating the simulation results is essential to explain the execution process and simulation results obtained. Animation is based on the xml output file obtained after all simulation instances had terminated. The purpose of the animation implemented in the SLMToolBox is to highlight the paths executed during the simulation, the sequence of Models which were executed, and the transitions from one state to another. The animation is based on changing colors of graphical objects. Figure 50 presents a simplified algorithm for developing the animation feature. Figure 51 is an extraction of the animation feature, it presents a step-by-step animation.

- Step 1 is the diagram at its initial state before starting the animation
- Step 2 is the beginning of the animation where the first model and its active state are highlighted.
- Step 3 reveals active states and model which are active after the execution of an internal transition in the model capture incident and an external transition in the classify incident.

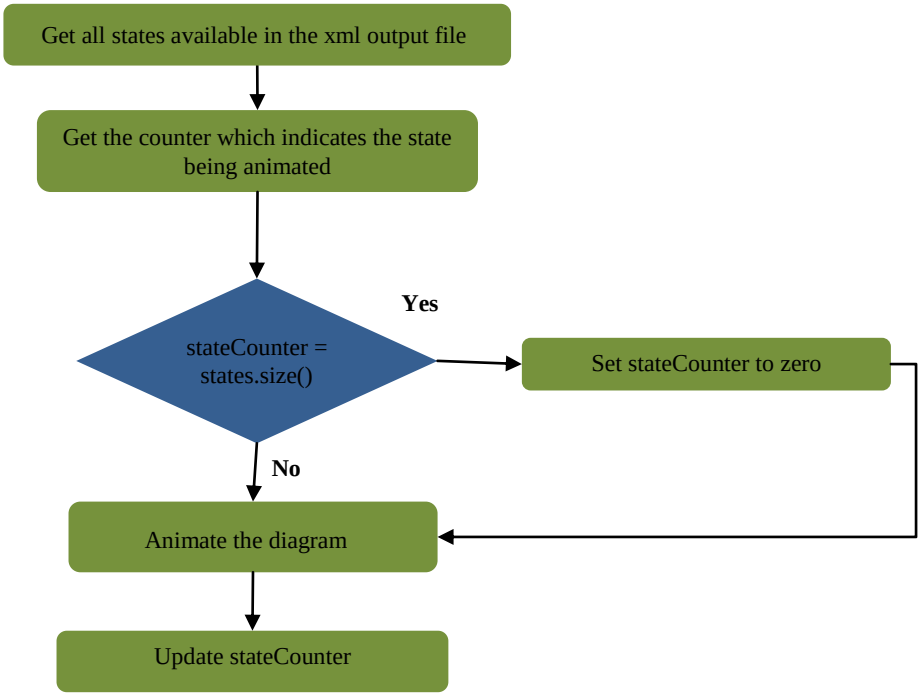


Figure 48 Animation simplified algorithm

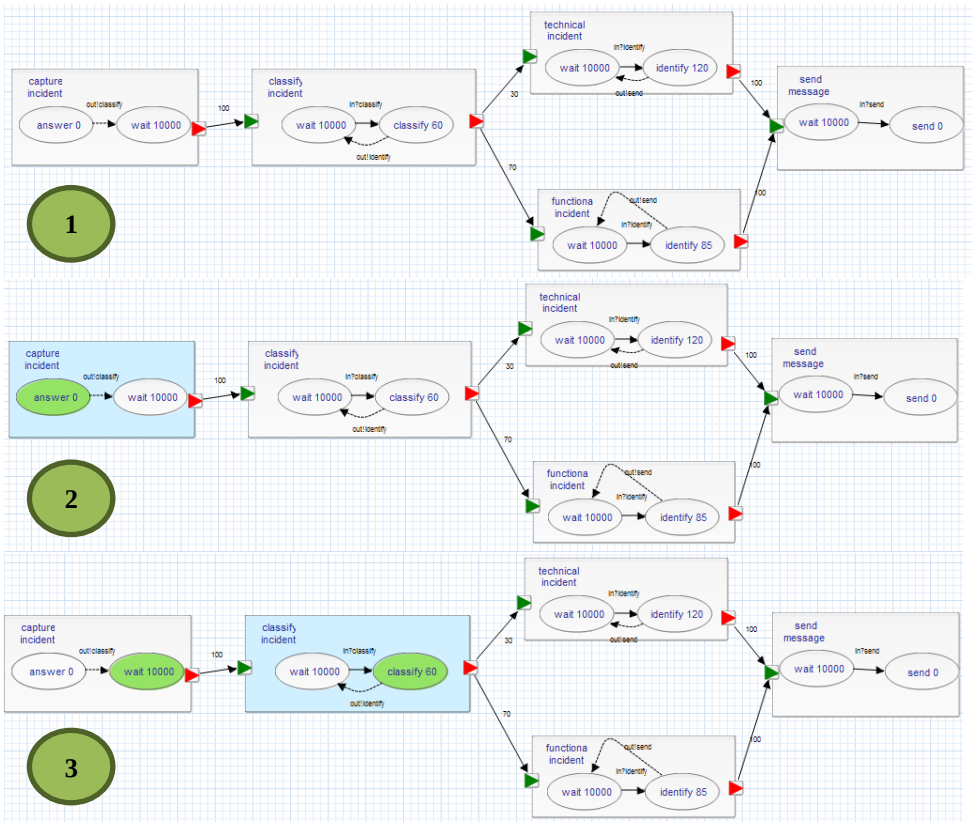


Figure 49 Animation feature

6. Example

In section 3.3.4 we presented one of the processes value chains realized within a collaborative e-marketplace network. The process was first modeled at BSM level using the EA* modeling language and then it was transformed to BPMN model. In this section, we will transform the obtained BPMN model into a DEVS model. The goal of this transformation is to simulate this collaborative process in order to obtain more information on time needed for a user to browse, choose, and buy a product using the e-marketplace. The simulation results will help business engineers and analysts study better the user's attitude. Different paths taken by the user can be studied which can lead to the modification of certain features to facilitate the user's surfing through the e-marketplace

The BPMN to DEVS transformation feature implemented in the SLMToolBox is based on a simplified mapping. As a result of this transformation, the DEVS diagram obtained after using the SLMToolBox is represented in figure 52. Several transformation rules are not respected in this simplified implementation and to be developed in future extensions of the SLMToolBox. In the figure we can find that BPMN tasks of different types and gateways are all transformed to Atomic DEVS Models formed of two states. In addition BPMN lanes are not transformed or implemented in the transformation.

The obtained DEVS diagram needs to be enriched by the user before starting the simulation. The inputs to be added by the user are the following:

- Processing time of states in DEVS Atomic Models
- Cost of DEVS Atomic Models

- Probability of divergent internal couplings (by default the probability is 100)
- Associated values and ports for External and Internal Transitions

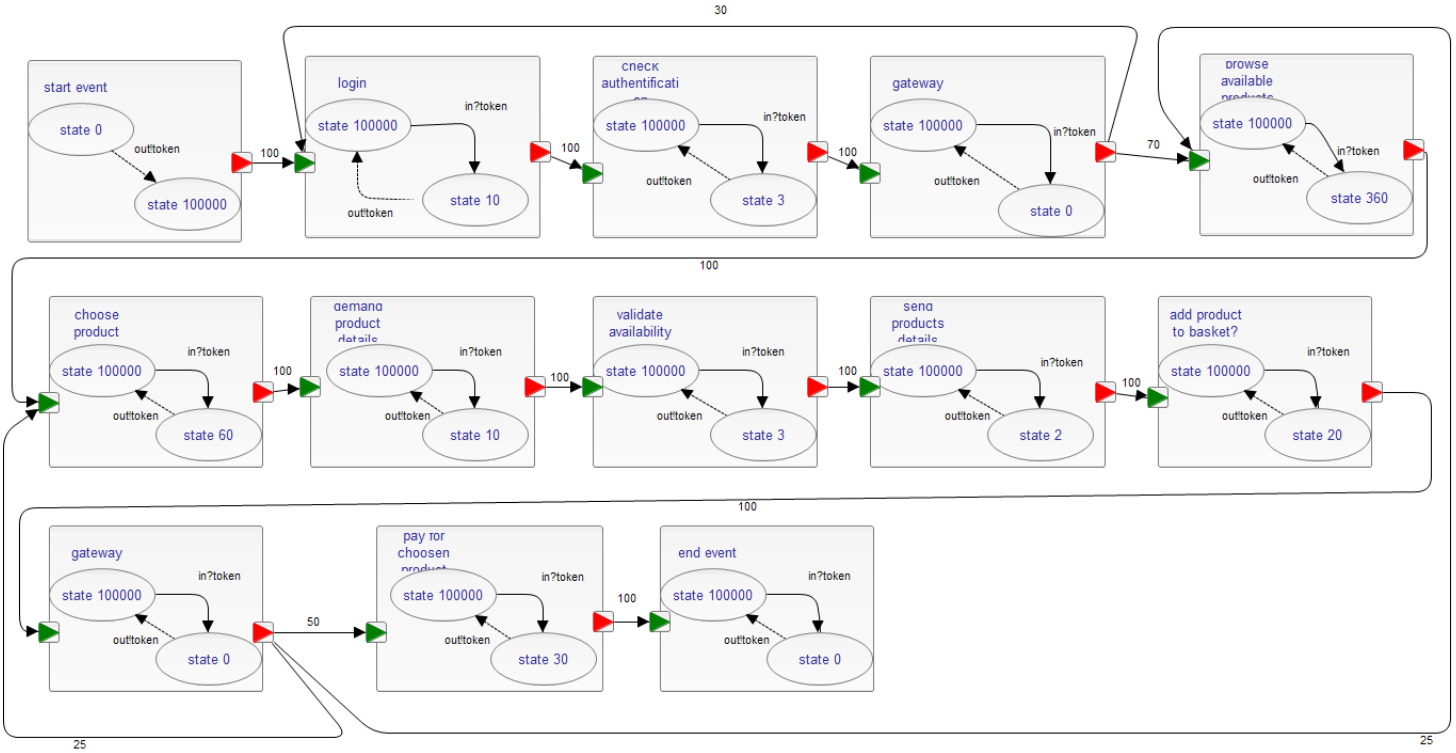


Figure 50 DEVS e-marketplace purchase process

After enriching the DEVS diagram the user is supposed to initialize the simulation profile as explained in the figure 53. In our example we entered 100 as number of instances and 600 seconds the time for a user to choose and buy his product (objective). The cost is not assigned since the reason of the simulation is to study user's behavior and the time spent throughout the process. Now the user can start simulating the process and a report in pdf format is created. The content of simulation reports obtained after the execution of the simulation is tib explained later in chapter 5.

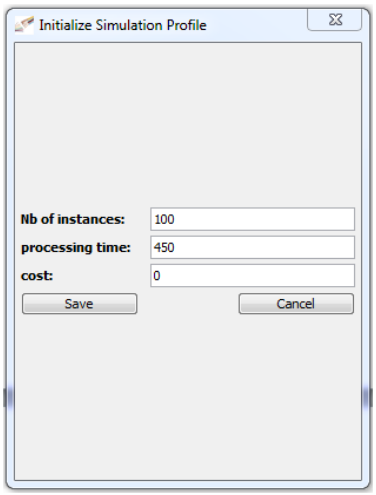


Figure 51 initialize simulation profile

7. Conclusion

In this chapter we presented DEVS to simulate service's behavior and our contribution in the development of a **DEVS editor and simulator** for business processes in service systems. We started by defining the problem and the need for a **model transformation** from BPMN models to DEVS models. We illustrated an introduction on the characteristics of the DEVS formalism and the simulation of DEVS models. Then the transformation from BPMN to DEVS models is detailed and the mapping is defined. In addition, we presented our work on the simulation of DEVS models by defining the **execution, profiles and results, and animation of simulation results**. At the end of this chapter an example of transformation from BPMN to DEVS models and the simulation of the obtained result are illustrated.

In the previous chapters we talked about methodologies and theoretical results of work. The next chapter presents the implementation part of this thesis, the **SLMToolBox** as a modeling and simulation tool.

SLMToolBox

Chapters 3 and 4 presented the MDSEA methodology, EA modeling language, and transformation and simulation concepts in MDSEA. This chapter will present the SLMToolBox as the applicative part of this thesis and a validation of the theoretical part. SLMToolBox (Service Lifecycle Management Tool Box) is a software tool which supports an organization to engineer new services or improve existing ones and to manage its life cycle. The SLMToolBox is a modelling environment dedicated to the domain of service engineering. It is based on the Model Driven Service Engineering Architecture (MDSEA) concepts and supports the first phases of service engineering, in particular: service requirement and service design. The software is developed in the frame of the IP European Project "MSEE".*

1. System overview

1.1 Context and purpose

The objective of the SLMToolBox is to support the phases related to service engineering, within the "service lifecycle" model. It is important to make a clear difference between the "context" in which the SLMToolBox is used and the phases of the service lifecycle it aims to support. SLMToolBox will be used in the frame of enterprise projects which aim at developing a new service or an improvement on a service, within an organization (composed either by one single enterprise; or by several partners, in this case: a virtual manufacturing enterprise). The tool will be used at the stage of "requirement" and "design" (figure 54) of the service engineering process.

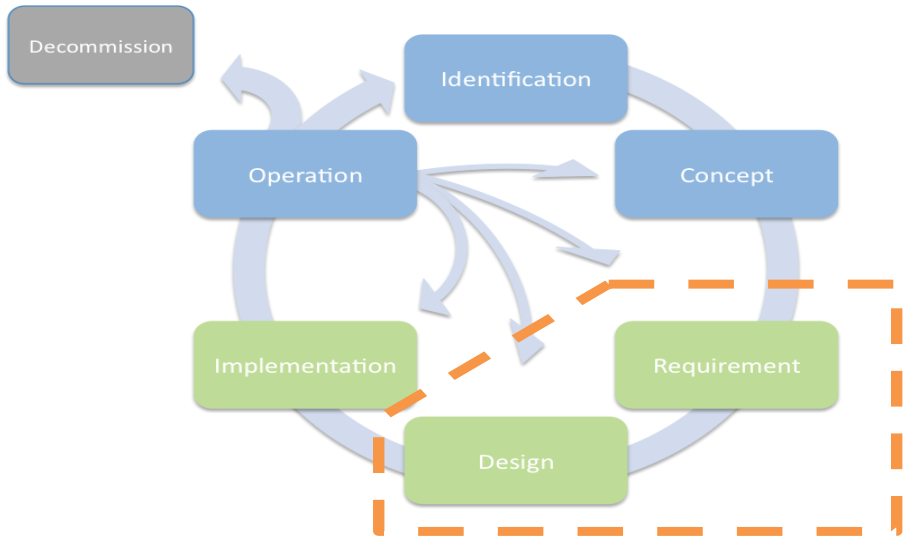


Figure 52 SLMToolBox - Context within the service lifecycle

During the requirement and design stage of the service, the tool will be used to describe in details "how the service will behave" in the operation phase of its lifecycle (figure 54). As a complement, it is possible to also describe the next phases of the service in its lifecycle:

- How (with which process /resources / tools) will the service be designed; implemented?
- How the service will be decommissioned at the end of its lifecycle?

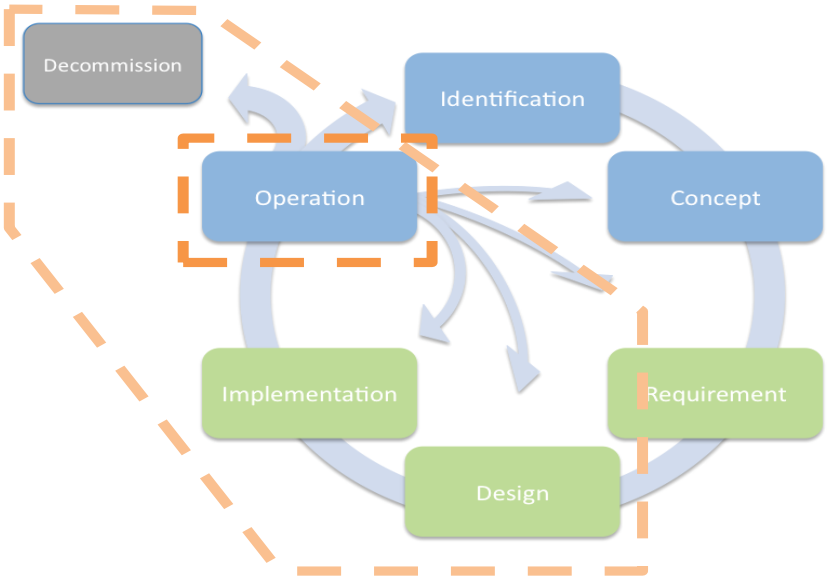


Figure 53 Purpose: phases of the service lifecycle to support

1.2 System vision and top level requirements

The main motivation for the development of the SLMToolBox could be formulated as the following: "no reference tool for designing and managing service innovation projects (Servitization process) currently exists. It affects European Manufacturers willing to invest on service innovation: as they currently have to rely on various generic tools, mostly oriented on business process management and software engineering domain".
Stake holders willing to create or modify a service within an organization (either a single enterprise or a virtual manufacturing enterprise) require:

- to specify, evaluate, communicate and design the system supporting the service and its lifecycle
- appropriate formalisms (domain specific & easy to read)
- productive means ; Interoperable formats

In addition development teams attempting an optimized development of the IT part of a service system (example: an online shirt configurator) need to:

- elaborate a solution which is directly connected to the initial requirements (e.g. : integrates with the business processes of the company)
- concentrate on technical activities (e.g.: technical design, implementation …)

SLMToolBox is an integrated modelling tool, dedicated to manufacturing services lifecycle management which will allow to:

- take benefit of a model based architecture: syntactic validation, transformation, execution…
- maintain the coherence through the whole engineering process - from Business requirements to IT implementation (modelling)
- anticipate / simulate the result of the service (engineering)
- design the governance of the service (monitoring & control)

Unlike other CASE tools (e.g.: UML modelling Tools, Business Process Management Tools …) it will guide the development of new services and service systems in a coherent approach, from the business perspective, to the design perspective.

The SLMToolBox has certain limitations since it does not provide support for: implementation / coding software components, implementation of Business Intelligence report, neither monitoring of service's execution. The flexibility of the modelling architecture is limited to the instantiation of the metamodels which are linked to the software at the development time. It means that the structure and the template available to model services are static. Therefore, a modification on the modelling languages and metamodels will necessitate new coding activities through a new development phase.

1.3 Logical architecture

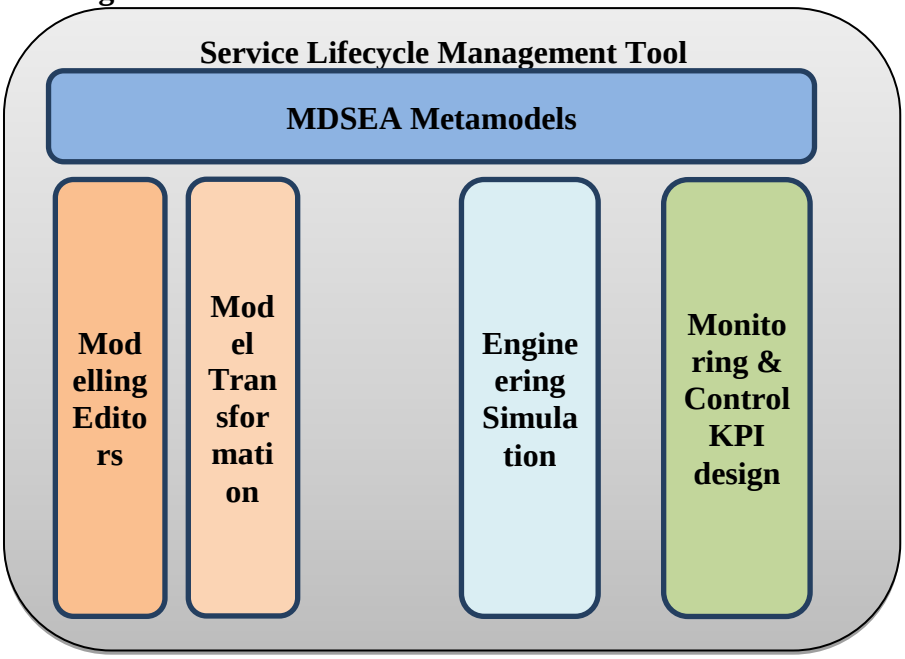


Figure 54 SLMToolBox Logical Architecture

The foundation of the SLMToolBox is based on the MDSEA modelling architecture. This model centric approach provides the appropriate structure for elaborating service requirement and design thanks to a set of specific metamodels – dedicated to the domain of manufacturing services.
The first pillar of the architecture brings a set of modelling editors, enabling the user to elaborate structured and graphical descriptions of the service and its aspects (IT, Human, and Physical Means) – at the business level (BSM : Business Service Models) and the design level (TIM : Technology Independent Models). As a complement, model transformation facilities will leverage interoperability of the models and enforce consistence between the Business requirements of the service and its design at TIM level.

The second pillar aims at sustaining the modelling activities thanks to a methodological support. Guidance will be provided to the user through the modelling activities of the service via an appropriate service engineering methodology. Besides, some support will be provided to assess the overall quality of the service at high level – at design time, thanks to appropriate tools.
The third pillar is responsible for the simulation of business processes providing animation and simulation reports (chapter 5).
The fourth pillar will support the definition of the service system's governance, which will be then implemented by the organization to continuously assess the performance of the service according to the three decision levels of the organization (Strategic, Tactical, and Operational), its functions and its detailed objectives.

1.4 Actors and roles

The MSEE IT System will provide several functionalities that will be used by different users. In this section the roles of the actors involved in the activities of service requirement and service design phases will be identified and described. This list is partially derived from the actors identified at the level of the generic MSEE IT architecture. The roles describe bellow define at a conceptual level the categories of user profiles concerned by the SLMToolBox features. From the two modelling abstraction levels covered by the SLMToolBox (BSM and TIM), it is trivial to derive two categories of actors which will contribute to the modelling activities, in interaction with the Modelling Environment, provided by the SLMToolBox.

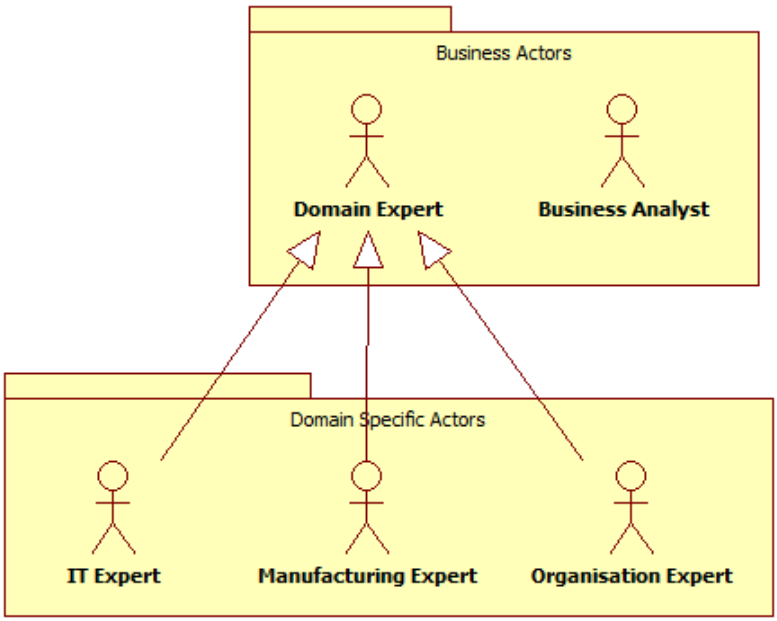


Figure 55 System Actors

1.4.1 Business Actors

This category includes the actors which can collect the knowledge of the enterprise at the business level.

- **Business Analyst**: is the actor that can collect the knowledge and the requirements at the highest level of the enterprise. He is interested in the analysis of its enterprise / ecosystem and the development of the service system at a global level.

- **Domain expert**: this actor is an expert of a specific domain inside the enterprise / ecosystem. This can be one of the following: IT, Manufacturing, Organization. The domain expert is able to bring specific knowledge and constraints related to its specific domain. He can identify the impact of the servitization at an operational level and proceed to the design of the modifications to implement in its domain.

1.4.2 Domain Specific Actors

The "domain specific" modelling activities at TIM level will be handled or in the responsibility of the corresponding domain actors: IT, Manufacturing, and Organization. This category defines the actors related to one of the three specific domains of the service system.

- **IT Expert**: this actor can collect the knowledge and proceed to the design related to the IT system of the enterprise / ecosystem (including: infrastructure, applications, data repositories …).

- **Manufacturing Expert**: this actor can collect the knowledge and proceed to the design related to the physical means of the enterprise / ecosystem (including: manufacturing machines, supply chain, products design …)

- **Organization Expert**: this actor can collect the knowledge and proceed to the design related to the organizational aspects of the enterprise / ecosystem (including: human resources …).

1.5 End-to-end scenarios

The SLMToolBox essentially supports the "requirement" and "design" phases of service engineering. The two following subsections illustrate two major scenarios, involving the main features of the SLMToolBox, through specific use cases and their sequence.

1.5.1 Scenario 1: Design a new service within a single enterprise

This first scenario depicts how the SLMToolBox will be used to design a new service, within a single enterprise.

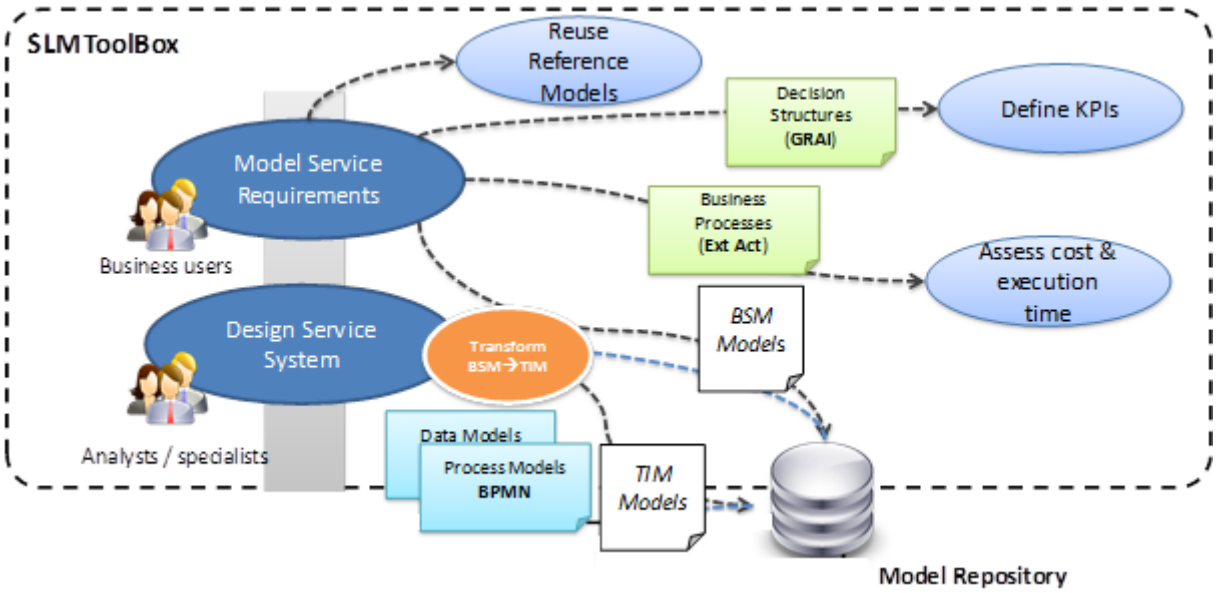


Figure 56 Design a new service within a single enterprise

This scenario is driven by three main use cases:

1. **Model service requirement (supported by the SLMToolBox – BSM modelling features)**
    1. Reuse reference models : the business user has the possibility to browse the model repository and search for a convenient reference model to start modelling the service requirements in a BSM modelling project
    2. A BSM model is initialized and enriched trough the template editor (for generic service description) and extended with graphical models; the BSM models are stored within the model repository, shared with the rest of the MSEE IT system. The overall modelling process at BSM level follow the "BSM Service Modelling" method, derived from D11.2 – "Service concepts, models and method: Model Driven Service Engineering"
    3. The governance system of the service is modelled through the GraiGrid editor
    4. The KPIs of the service are defined on the basis of the GRAI grid model
    5. Business process are elaborated with the Extended Actigram Star language
    6. Some of these processes can then be simulated in order to assess their execution time and cost
2. **Design service system (supported by the SLMToolBox – TIM modelling features)**
    1. The first step of the design phase is to retrieve the BSM models from the model repository and to initialize a TIM modelling project, thanks to automatic model transformation techniques
    2. A TIM model is initialized and enriched trough the template editor (for generic service description) and extended with graphical models ; the BSM models are stored within the model repository, shared with the rest of the MSEE IT system
    3. UML models are elaborated via the UML modeler
    4. Extended Actigram star process models from the BSM modelling project can be automatically transformed into BPMN process models, either "collaboration diagram" or "process models". The resulting BPMN models are attached to the current TIM modelling project
    5. BPMN process models can be modified / enriched by the user, within the TIM modelling project

1.5.2 Scenario 2: Design & deploy a new service within a VME

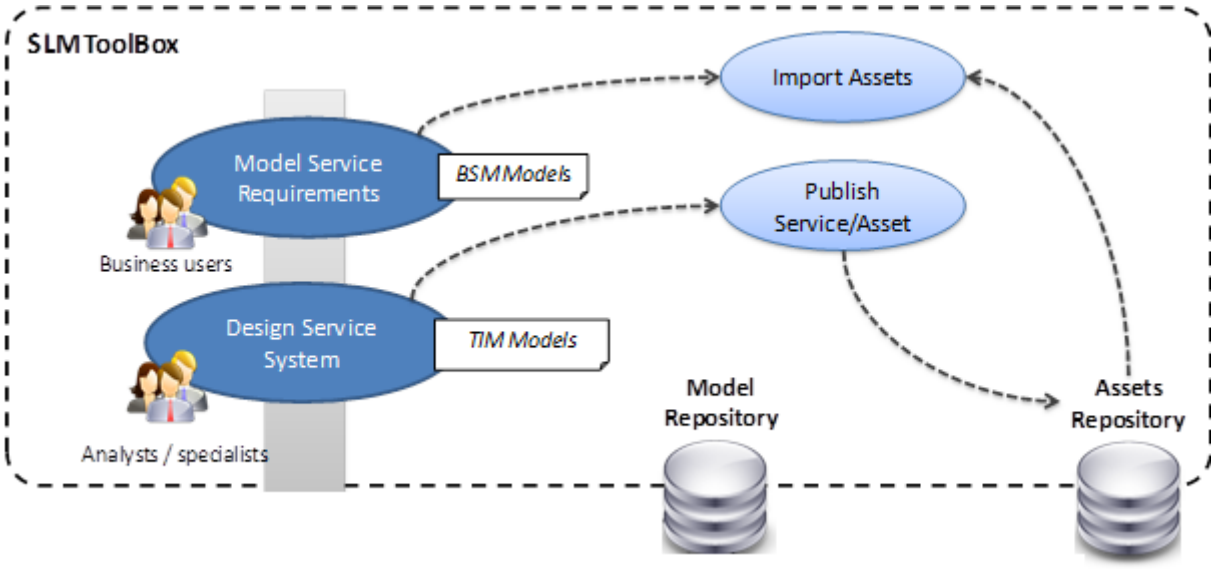


Figure 57 Design & deploy a new service within a VME

This scenario is driven by three main use cases:

1. **Model service requirement (supported by the SLMToolBox – BSM modelling features)**
    1. The modelling activities at the BSM level are similar to the previous scenario. However, in the context of a virtual enterprise, we assume that the modeller should be able to retrieve the description of the assets of his partners. In practice, the user of the SLMToolBox is able to connect to the Assets Repository, to browse and search for relevant assets to include in its service models; so that he is able to "compose" a new service, on the basis of existing assets exposed by the members of the VME.
2. **Model service requirement (supported by the SLMToolBox – BSM modelling features)**
    1. The modelling activities at the TIM level are similar to the previous scenario. However, in the context of a virtual enterprise, we assume that the modeller should be able to publish the description of the service having been modelled; so that the VME is now aware of the characteristics of the new service being developed. In practice, we propose that the modelling activities of the "virtualization process" for tangible / intangible assets would be supported by the SLMToolBox, until the assets description would be published on the Assets repository.
    2. Operational processes are modelled with the BPMN editor of the SLMToolBox

2. Technical overview

Figure 60 gives an overview of the several technical components that compose the modelling environment of the SLMToolBox. We differentiate the "application components" which are specifically implemented and are part of the domain of the service system modelling tool, from the "technical components" which refers to existing development artifacts, like framework, libraries and APIs. These technical components are uses as the basic building blocks of the application and are issued from the technical analysis described in the previous section.

2.1 Technical modules

Eclipse Platform

The Eclipse Platform remains the main technical foundation for the SLMToolBox environment. Considering its background in research projects, the large community supporting the development of the core platform and its rich ecosystem of plugins, the Eclipse Platform [eclipse] is considered as one of the most viable open source solutions for building domain specific modelling environments [Amyot et al, 2006].

EMF

The Eclipse Modelling Framework – EMF [EMF] provides a modelling infrastructure for describing metamodels and editing models with the help of Ecore format and code generation facilities. Furthermore, EMF is used as a foundation by numerous eclipse projects, which address different aspects modelling activities (transformation, persistence, editing, visualization …) and that can provide good support for the implementation of the main features of the modelling environment.

EEF

While EMF natively provides basic editing facilities for Ecore models, the Extended Editing Framework – EEF [EEF] aims at providing new services dedicated to editing and using more appealing editing elements for EMF models. As EMF, EEF relies on a generative approach to provide advanced editing services. This approach is particularly suited for domain specific metamodels which do not define graphical formalisms to represent models. In the case of

MSEE, BSM and TIM metamodels are defined as the specific core of service system modelling, and need to be editable via a rich interface, while no graphical formalism is designed for the representation of BSM and TIM models. Thus, we propose to provide a set of editing features, allowing visualizing the BSM and TIM models under the form of a tree view and a set of forms to edit their structure and attributes. In this case, EEF performs as a good candidate, to provide specific editing features for BSM and TIM model constructs.

Graphiti

As presented and evaluated in the previous section, Graphiti [Graphiti] offers powerful means for building graphical diagrams editors upon EMF based domain models. Graphiti provides a set of common user oriented features "out of the box" such as diagram layout, undo/redo actions, keyboard shortcuts handling, rich graphical object design ; which allow the developer to focus on domain specific code. Furthermore, it provides convenient extension points to integrate Graphiti editors in a large Eclipse application.

Model repository integration

The modelling environment must offer storage capabilities in order to persist models along their lifecycle, and to allow the capability to retrieve them and to update them. Moreover, as some of the models (at TIM level) will be shared with the Generic Service Development platform, the models should be persisted in a central repository accessed via both systems. Finally, as multiple instances of the SLMToolBox and the Service Development Platform may access the same models in a collaborative way, this repository has to handle cases such as concurrent access and editing conflict resolution. The SLMToolBox will integrate the "Model Repository" client component, which provides access to the MSEE model repository.

Assets repository integration

The access to the assets repository will be managed through a client plugin component, integrated to the modelling environment of the SLMToolBox. Connection, browse, search, retrieve, and publish actions will be managed through the manipulation of the REST API of the assets repository via the client plugin.

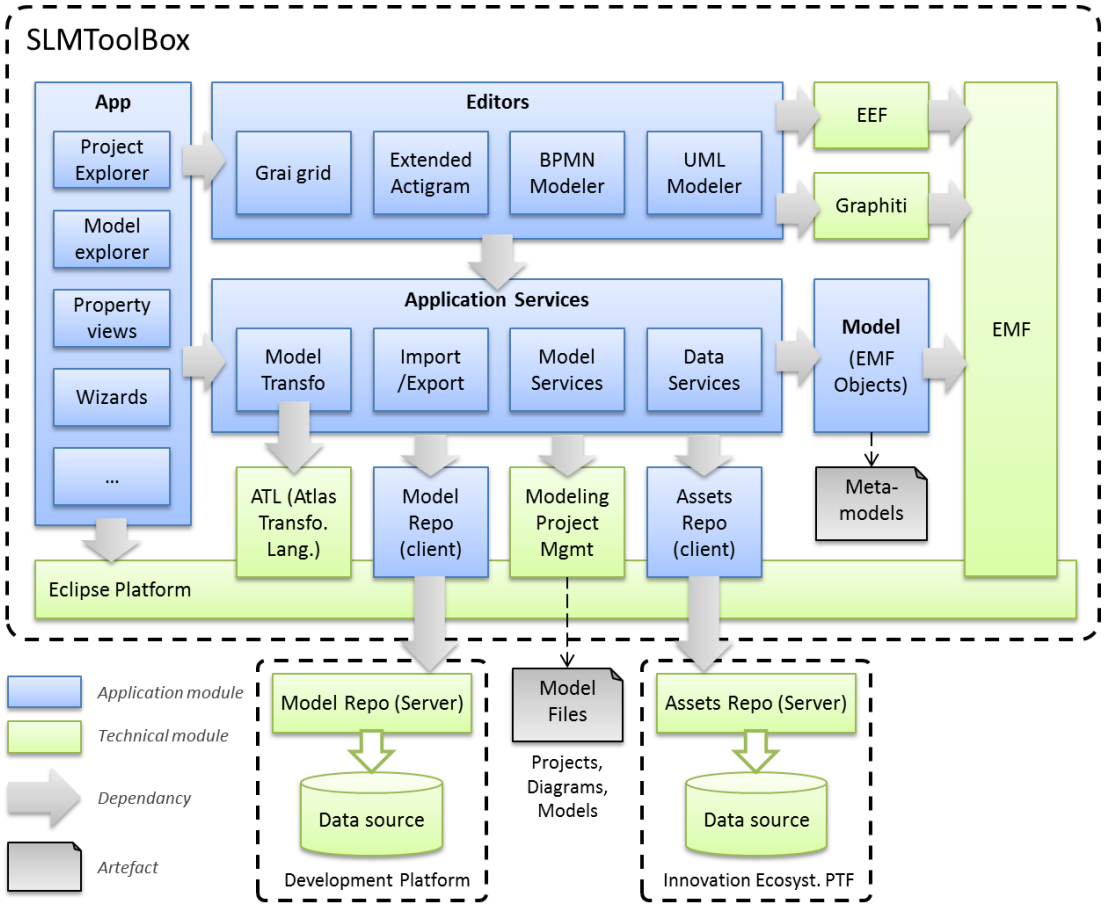


Figure 58 Modelling Environment - Technical Architecture Overview

2.2 Application modules

App module

The application module is the main application container of the modelling environment of the SLMToolBox. It is mainly responsible to compose and provide the graphical interface to the user under the form of several views, being used to navigate through the modelling project content and edit service system models. It coordinates the execution of the underlying features of the modelling environment, and contains the logic of the graphical presentation of the overall application. To fulfill its role, this module heavily relies on the eclipse core platform API, to benefit from the artifacts it provides for standalone applications (views, explorers, wizards …).

Model objects module

This module encompasses the definition of the model objects of the domain of the application and their interrelationships. It provides an implementation of the conceptual model of the application under the form of a set of java beans. This implementation is generated with the EMF code generation features, from the definition of the metamodels in Ecore format at the development time. This module plays a central role in the application as it provides the objects instances to the other modules, responsible for processing, presenting and persisting these objects.

Editors module

This module is responsible for providing independent graphical modelling editors to the

application. Two types of editors are provided:

- Graphical editors allow the editing of diagrams with the help of graphical elements, related to a specific modelling language (for instance: ExtendedActigramStar). While the diagrams are edited, the editor stores the diagram in a specific file and delegates the persistence of the model data to the application service module. Each editor relies on the Graphiti framework, in order to provide standard editing facilities and to offer a rich set of graphical elements to the user.
- The second set of editors is designed for domain specific purpose and allows editing BSM and TIM models. To deserve this goal, a tree view is provided to browse the model content, and a set of property sheets to edit the attributes of the model objects. As for the graphical editors, the persistence and the update of the model data is delegated to the appropriate service in the application service module.

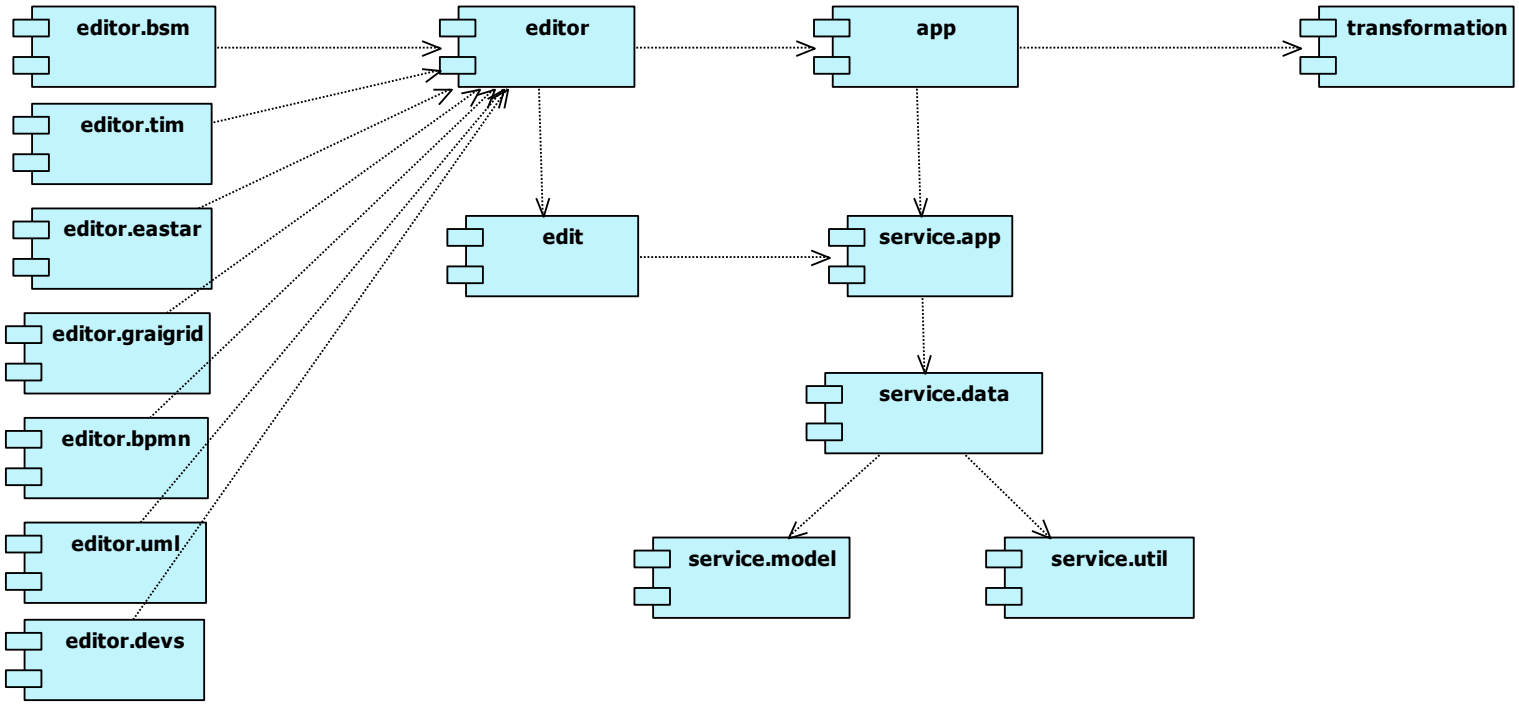


Figure 59 Application modules

Application services module

This module is responsible to handle the domain logic of the application, which is independent from the presentation mechanisms and from the data storage features. It is logically decomposed in four sub components:

- The model transformation component packages the transformation rules that apply to the domain of service system modelling, along MDSEA principles. It relies on ATL to provide model to model transformation routines and exposes its features to the application module.
- The Import/Export component is responsible to handle the processes that are necessary for importing standard models (example: BPMN models) in a service system model project and exporting models in standard representations (example: USDL models).

For specific logic, related to the transformation of MDSEA models to a standard representation, this component relies on the model transformation component.

- The Model Service component provides the basic services that are needed by the editors, model transformations and import/export components to manipulate model objects. This component acts as a façade and provides a unified interface to retrieve, check, modify and create MDSEA model subsets.

3. Implementation of MDSEA in the ToolBox

3.1 Modelling architecture overview

MDSEA defines a set of constructs and relationships (described with "templates") which are specific to the domain of service system modelling at three modelling levels: BSM, TIM, and TSM. For each abstraction level, MDSEA suggest a set of graphical modelling languages (which are domain agnostic) in order to extend and complete the representation of the system to be modelled under different perspectives (e.g.: decision structure, process, use cases…).
This type of modelling architecture is based on a "view model" pattern (or "viewpoints framework") [ISO/IEC/IEEE 42010 2011], Systems and software engineering — Architecture description) as it defines a coherent set of views to be used in the construction of a manufacturing service. The purpose of views and viewpoints is to enable humans to comprehend very complex systems, to organize the elements of the problem and the solution around domains of expertise, and to separate concerns. In the engineering of physically intensive systems, viewpoints often correspond to capabilities and responsibilities within the engineering organization. Both BSM (Business Service Models) and TIM (Technology Independent Models) are structured in the same manner. A "core" model gathers a set of generic (meta-) data in order to qualify the service to be modelled (specified / designed) ; this "core" model refers to external graphical modelling languages (e.g. : UML) so that certain aspects of the service model can be elaborated in more details with the help of graphical languages. This structure allows to map "view specific" modelling languages (e.g.: GraiGrid, UML Class Diagram) with "domain specific" constructs (i.e.: MDSEA BSM) without introducing modifications or restrictions to the MDSEA metamodel. From the user point of view, it allows the possibility to edit core information, independent from any specific modelling language, and to retrieve and reuse this data under different views, accomplished with the help of several graphical diagrams.

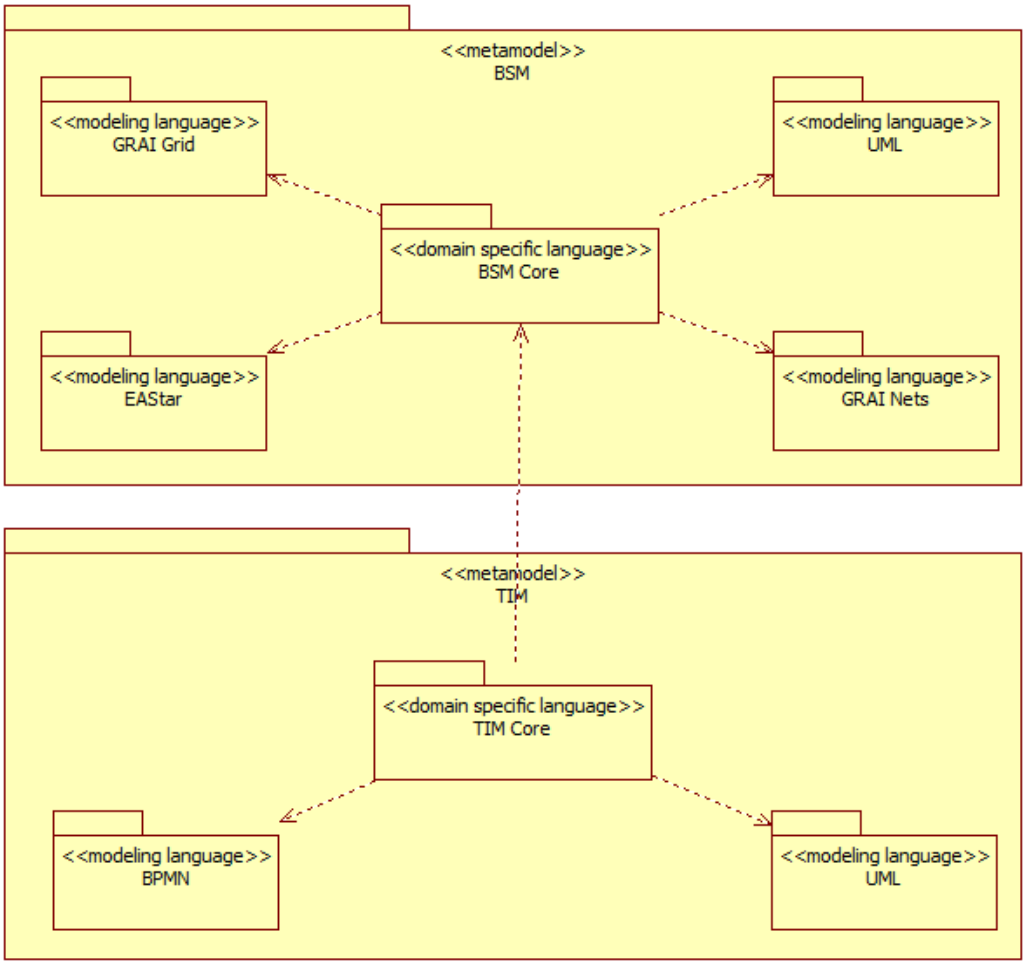


Figure 60 Modelling architecture's overview

With this approach, MDSEA Core Constructs remain agnostic from any representation formalism. Their implementation is realized by a core model, which acts as domain specific (Service System Modelling) "glue" between several modelling languages. Thus, we can reuse standard modelling languages without introducing modifications to their metamodel (e.g.: BPMN, UML…). Graphical languages such as "ExtendedActigramStar" or "GraiGrid" can continue to evolve, with (almost) no impact on MDSEA Core metamodels (i.e.: BSM).

3.2 Service modelling features

3.2.1 Summary of modelling editors

The modelling environment supports the service system modelling activities by providing editors for domain specific models (BSM, TIM) and related modelling languages to enhance the description of the BSM and TIM models. In our functional approach, we propose to provide a set of language specific modelling editors for each modelling language. The following table gives an overview of the modelling editors to be included in the SLMToolBox for each modelling level (BSM and TIM). These modelling editors are integrated within the same environment and technical platform (Eclipse Juno) in order to maintain data interoperability; coherence between models and improve the usability of the tool, from the user perspective.

Table 50 SLMToolBox - Modelling editor's overview

Modelling Level	Goal	Modelling Language	Editor
BSM	Describe service at high level	BSM Templates	Specific Development
BSM	Describe simple business processes	Extended Actigram Star	Specific Development
BSM	Describe decisional structures of the organization	Grai Grid	Specific Development
BSM	Describe Information Structures	UML (Use Case, Class Diagrams…)	Open Source Plugin (PAPYRUS)
TIM	Describe service at high level	TIM Templates	Specific Development
TIM	Describe detailed business processes	BPMN2.0	Open Source Plugin (BPMN2.0 Modeler)
TIM	Specify the IT artefacts	UML (Use Case, Class Diagrams …)	Open Source Plugin (PAPYRUS)

3.2.2 **GraiGrid Editor**

A virtual manufacturing enterprise (VME) is a temporary alliance of companies for the lifetime of a joint production of service. VMEs are such entities, which, from the point of view of their service to the customer, appears to be one entity, but in reality are formed from several autonomous entities, or partners. The property that differentiates a virtual enterprise from an ordinary value chain is the fact that there is a single locus, which takes full responsibility for the entire value chain of its product or products, even though the task is carried out by many participants and for that reason they cooperation must be harmonic.

The GRAI Grid modelling language is used for modelling the decisional structure of the specific enterprise. The GRAI grid concept relies on the fact that any management decision that needs to be taken will always be made with reference to a specific time horizon. Managers typically define strategic, tactical, operational and real-time management levels. These levels implicitly involve a hierarchy of decision functions structured according to decision horizons or periods. These cells represent decision centers which can have two types of connections: non-hierarchical and hierarchical connections. In a VME, the use of the Grai-Grid allows to represent decisions concerning product and resource management and planning in various enterprise entities. For this purpose, we introduce a new concept we call "Collaborative Grai Grids". These concepts permits to merge (combine) the Grai Grid of each partner in order to provide a whole Grai Grid for the Virtual Enterprise. A Virtual Manufacturing Enterprise (VME) integrates N manufacturing enterprises. The decisional structure of each manufacturing enterprise is defined by a Grai Grid. In order to elaborate the decisional structure of the VME, we propose to combine and to structure several grids. The user of the SLMToolBox at BSM level is usually aware of the different decision structures (Grai Grid) that belong to the VME partners and the dependencies between them. As a result he is able to model this collaboration in a one combined Grai Grid.

In addition to modelling the decisional view of an organization or collaboration in a VME, with GraiGrid editor the user is able to define the:

- objectives which are associated to the decision frames of the piloting system,

- decision variables as drivers on which the decisions can act to reach the « objectives »
- primary indicators [Carosi et al 2014] as quantifiable and measurable data which measure the efficiency of an activity or a set of activity

The user will formalize these definitions on the basis of the GraiGrid modelling editor, which will allow enriching the BSM models with the data related to the governance model of the service system, select appropriate indicators from a reference list according to a set of search criteria's, and propose facilities to check the coherence (links and weights) of the triplets {objective, drivers – decision variables, and primary indicators} for each decision center.

3.2.3 **ExtendedActigramStar Editor (BSM Level)**

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. In order to model these relations and collaborations between partners, collaboration diagrams should be developed (were necessary) at the various abstraction levels of the MDSEA (BSM-TIM-TSM). The Extended Actigram Star language models business processes at the business level (BSM), it offers the concept of connectors (InternalConnectors, ExternalConnectors, and ProcessConnectors) which represents collaboration between entities within the same organization (single enterprise) or between different organizations (partners in a VME). In certain cases (collaboration between partners) users need a more presentable and readable presentation to demonstrate the collaboration.

The user of the SLMToolBox at BSM level is usually aware of the different processes that belong to the VME partners and the dependencies between them. As a result he is able to model this collaboration in a one detailed EA* diagram. Entities belonging to different organizations are differentiated using the organization concept introduced in EA* and implemented in the EA* editor.

In the same collaboration context the user of the SLMToolBox is able to connect to the Assets Repository, to browse and search for relevant assets to include in its service models, so that he is able to "compose" a new service, on the basis of existing assets exposed by the members of the VME.

3.2.4 **UML Editor**

Requirements: UML [OMG-1 2011] editing capabilities are required in order to capture the "domain model" at the BSM level and elaborate TIM models. The UML modeller must satisfy the following constraints:

- Integrate with the technical platform of the SLMToolBox (Eclipse Platform)
- Comply with UML2 standard XMI representation format
- Support the following UML diagram types: Use Case diagrams, Class Diagrams, Component Diagrams, Sequence Diagrams, and Activity Diagrams.

Integration of Papyrus: Papyrus is a dedicated tool for modelling within UML2; it is open source and based on the Eclipse environment. The key feature of Papyrus can be summarized as follow:

- Eclipse UML2 compliance
- Full respect of the UML2 standard as defined by the OMG
- Full respect of the DI2 (Diagram Interchange) [OMG, 2012] standard

- Extendable architecture of Papyrus [papyrus] that allows users to add new diagrams, new code generators, etc.
- Profile development support facilities for UML2 profiles

3.2.5 **BPMN Editor**

In MDSEA, the Business Process Management Notation (BPMN) is used for Business process modelling at the TIM level. A BPMN editor is required to be integrated in the SLMToolBox which can integrate with the eclipse platform, conforms to the BPMN specifications, and supply BPMN process and collaboration diagrams. The BPMN 2.0 Modeler provides an intuitive modelling tool for the business analyst, which conforms to well-established Eclipse user interface design practices. It also provides visual, graphical editing and creation of BPMN 2.0-compliant files with support for both the BPMN domain.

3.3 Model transformation features

The mapping of concepts proposed in previous chapters (4 and 5) is implemented using ATL (Atlas Transformation Language). Then XSLT (eXtensible Stylesheet Language Transformations) is used to create the graphical objects in order to open the transformed diagrams in their corresponding graphical editors (BPMN and DEVS editors). Using this combination of ATL and XSLT helps in separating the model concepts from graphical ones.

EA* to BPMN model transformation

Figure 63 presents the SLMToolBox creation wizard for the creation of new diagrams. User is able to create BPMN diagrams in two ways: either to start from scratches and creates a new bpmn diagram by the standard way, or to create a new diagram from an existing EA* one. The second choice requires a set of implementations in order to make it possible. After the user chooses the EA* diagram, an ATL transformation is applied which transforms the EA* model contained in the diagram into a BPMN model. Now that the BPMN model is available, it is important to generate its corresponding graphical objects. XSLT is used for such purpose and generated the diagram part of the model. The result of The XSLT transformation will be a BPMN diagram that can be opened using the BPMN modeler of the SLMToolBox. In annex 3 ATL and XSLT sample code are presented.

BPMN to DEVS model transformation

BPMN to DEVS transformation is implemented for simulation purposes. DEVS is the formalism used to study if the objectives identified by the user could be accomplished by business processes developed. The transformation from BPMN to DEVS is implemented and developed using same implementation strategy used for EA* to BPMN transformation. As BPMN diagrams DEVS diagrams can be created in two ways either from scratch or from an existing BPMN diagram. ATL and XSLT are used for obtaining a final DEVS diagram that can be viewed and simulated by the DEVS editor.

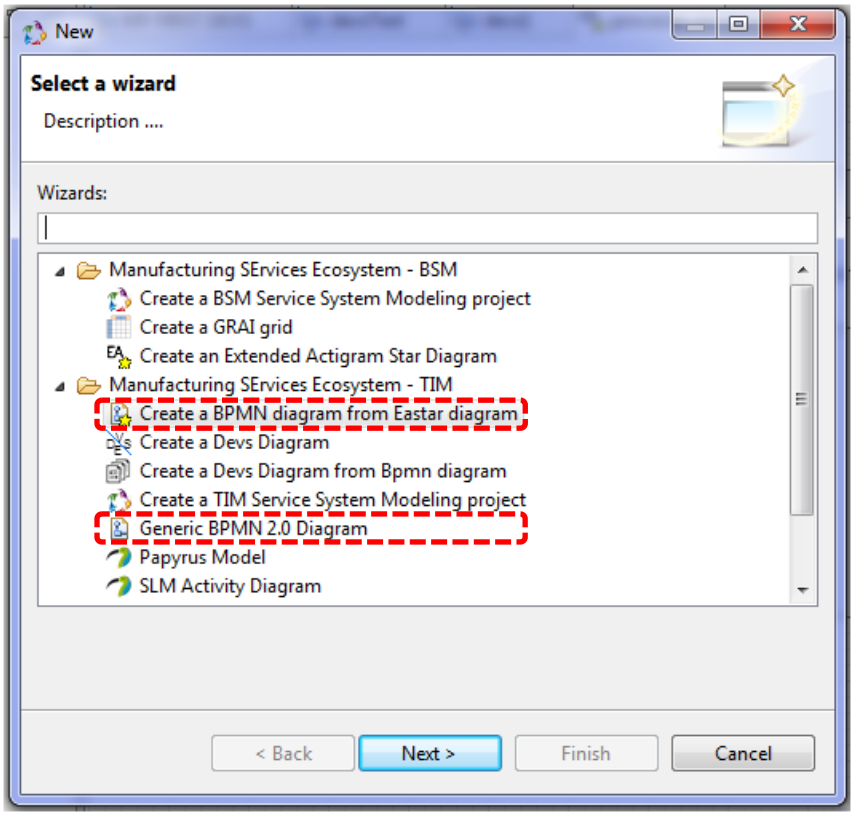


Figure 61 Create new wizard

4. Implementation of simulation in the ToolBox

The Toolbox possesses a simulation feature proposed in the frame of MDSEA. ToolBox Users are able to visualize devs models using a devs editor developed for this purpose, prepare models for simulation by defining a simulation's profile, simulate diagrams, animate diagrams based on simulation results, and provide a simulation report in pdf format.

4.1 DEVS Editor

The SLMToolBox possess a graphical editor based on the DEVS modelling language. For every DEVS concepts there exists a corresponding graphical object. Users are able to develop DEVS models from scratch or visualize a DEVS model resulting from the transformation of BPMN models to DEVS models. The figure below presents a DEVS model developed inside the SLMToolBox. DEVS concepts are available to the left of the editor in a palette of objects. The editor offers a hierarchical representation through the decomposition of coupled models into separate diagrams. Double clicking on a coupled model will open a new diagram which represents the coupled model.

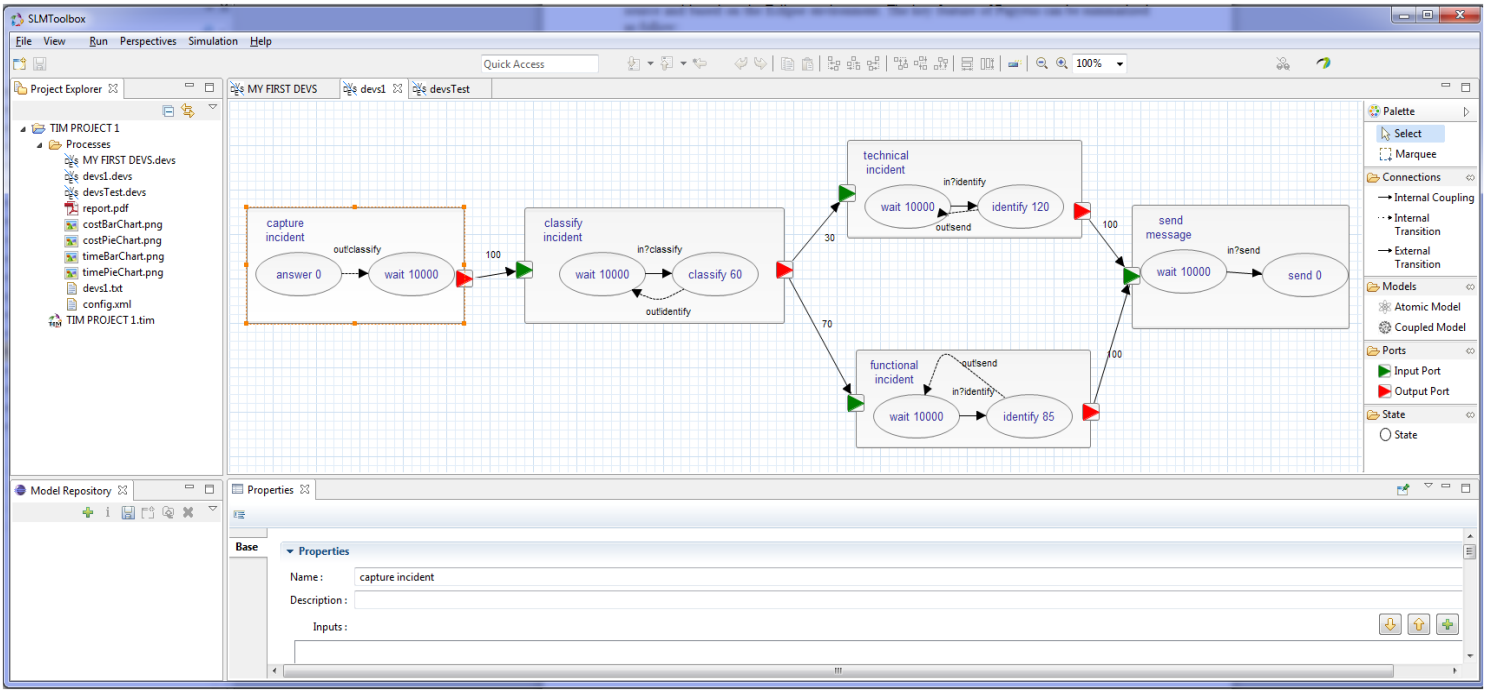


Figure 62 DEVS editor

4.2 Simulation profile

Simulation profile is necessary before starting any simulation. It defines basic aspects on which the simulation depends and results comparison also. Three aspects are defined with the simulation profile: number of instances to be simulated, user's processing time and cost waited by the user. The number of instances to be simulated signifies the number of times the DEVS simulation model is going to be executed starting from its entry point to its exit point (through its entire cycle). User's processing time and cost are the user's objectives at the end of the process. Based on these objectives and the obtained simulation results, users are able to proceed in the evaluation of their process.

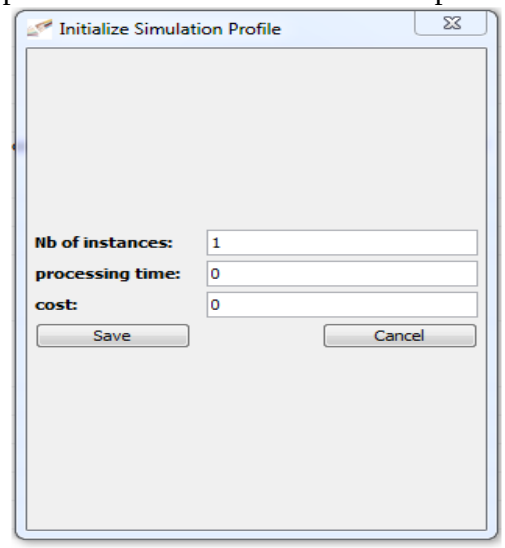


Figure 63 Simulation profile

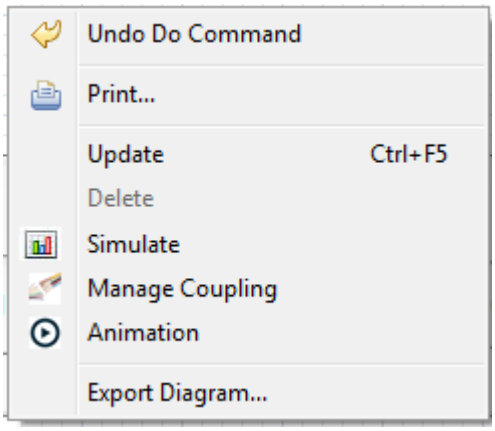


Figure 64 Simulation

4.3 Simulate DEVS model

The simulator is an implementation of classic DEVS. For the simulation to be well effected

the following requirements are demanded from the user:

- Initialize the simulation profile otherwise only one instance will be simulated, and the processing time and cost are set to zero.
- Set the cost of every atomic model, processing time of every state, and identify the active state of atomic models.
- Set the probabilities of internal couplings. In case of divergence, the path to be taken by the simulator is based on the probabilities of internal couplings emerging from the output port.

The user will choose to simulate as shown in figure 66. The simulator will run depending and on the inputs identified before. The simulator works on several hierarchical levels, starting from the first level which is the model to be simulated and the descending into the diagrams attached to coupled models.

4.4 Animate DEVS diagram

Animation of DEVS diagrams is based on the results obtained from the simulation. The animation indicates active states and models. “Ctrl + A” buttons are capable of showing a step by step animation, starting from the first active state and model till reaching the last active ones. Step by step animation will be realized by change of color as indicated in figure 67.

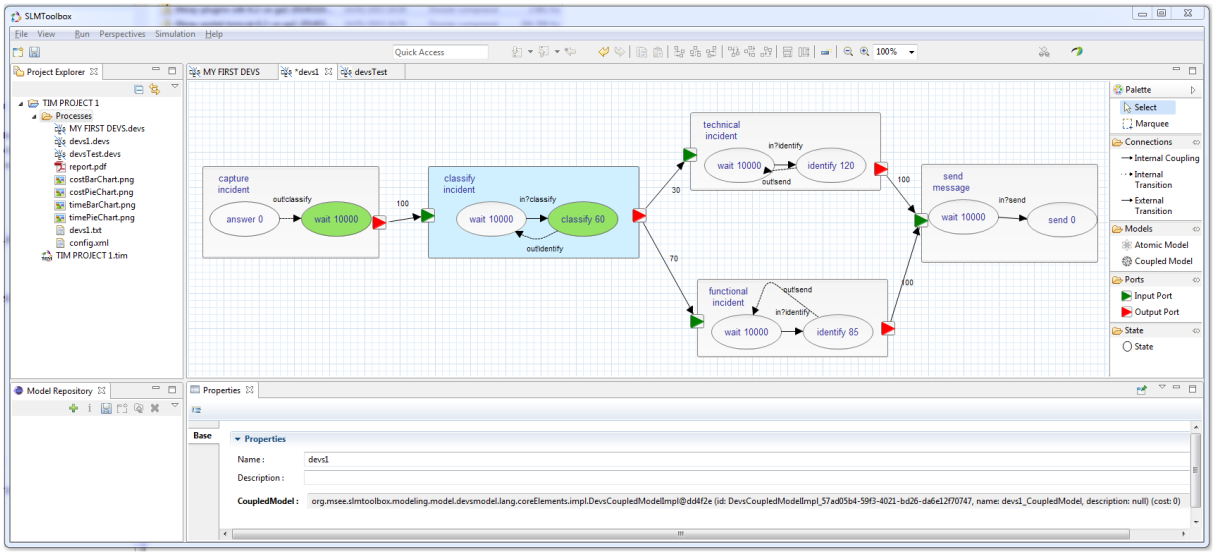


Figure 65 Animation

4.5 Simulation report

Annex 2 contains a simulation report produced by the SLMToolBox according to simulation results obtained. The report (pdf format) contains a listing of the simulation profile (number of instances and the user’s objective). Then information concerning the processing time are presented which includes:

- Pie chart and bar graph of different obtained processing times during the simulation
- Highest processing time and its corresponding path (sequence of atomic models)
- Lowest processing time with its corresponding path
- Most probable processing time with its corresponding path
- Least probable processing time with its corresponding path

Finally the report states the cost’s part which contains:

- Pie chart and bar graph of different obtained cost during the simulation
- Highest cost and its corresponding path (sequence of atomic models)
- Lowest cost with its corresponding path
- Most probable cost with its corresponding path
- Least probable cost with its corresponding path

5. Indesit Use case

Methodologies, methods, and tools that resulted from MSEE research and development activities are regarded as assets. These assets are used by the MSEE four industrial partners in order to help them in their transformation and shifting process towards servitization. The four industrial and manufacturer partners are regarded as use cases which have benefited from all MSEE assets in their servitization process and on the other side validated the usage and utility of these assets. Several assets have been developed during the MSEE project, but in the frame of this thesis only two assets are concerned: MDSEA and SLMToolBox. The first asset (MDSEA) aims at supporting business users and system engineers to model service-product and service systems along the SLM lifecycle. The structured approach is defined to guide the collecting of requirements, building models, design, and implementation. The second asset (SLMToolBox) will provide the graphical editors necessary to model manufacturing services and service systems from a "business perspective" (BSM) and a "functional perspective" (TIM) for service engineering activities.

In the following sections, the Indesit use case is presented with an overview of its AS-IS/TO-BE situations and the Carefree washing service to be produced in collaboration of other partners. Then the models resulting from using our two proposed assets (MDSEA & SLMToolBox) are presented.

5.1 The Use case experience: from Products to Services (AS-IS Situation)

The Indesit is actually the absolute leader in countries such as Italy, the United Kingdom and Russia. It was founded in 1975 and is listed on the Milan Stock Exchange since 1987; the company turnover for 2013 was 2.7 billion Euros. Headquarter has established in Fabiano (Italy), a town in Marche region where it leads over 300 after-sale centers in 150 cities. Actually Indesit has eight production sites (three in Italy, two in Poland and one in the United Kingdom, Russia and Turkey) and 16,000 employees, including over 4,000 in Italy. Also, Indesit established several commercial branches outside of Europe such as North and South America, Far East, Middle East and Africa as well. This enterprise has a deep consideration into the research and development of new products through spending around a third of investments. For instance, the number of patents registered is growing by an average of 30% a year, also over 600 people who work in this area (68% of whom are in Italy) reflects the strategy of the enterprise in order to be a pioneer in this field. Indesit has been established based on innovative appliance with technological solutions aim to do the housework in a smart and efficient way in order to enable the customers to simplify and enjoy their time. In this context, the selected enterprise represents the modern appliance with distinctive design not only to adapt with customer life-style but also to help them to make their home a uniquely rewarding experience through offering appliance ergonomic and silent running and intuitive performance.
All products are currently designed and commercialized in a traditional way as physical products. The product development cycle is strongly centred on product and the main

business processes are connected to the product stages (idea generation, feasibility, concept, design, development, testing). During the last years, almost 1.400 new product codes were created and innovation projects were product-oriented: 15 of them were focused on new aesthetics or functions and 11 of them interested aesthetical or functional upgrades.
Selling the physical product for Indesit means that the customers usually go to a retailer/distributor, see the product alternatives and choose the best one according to their needs. Usually the retailer is a big shop specialized in house appliances or domestic items and offer a wide variety of products, from mentioned enterprise and from its main competitors. Products are presented in large exhibition spaces and located in different layouts. The more common expositions use the grid-based layout or the round-based layout: the former allows customers to see all products in line and explore them throughout a fixed path; the latter creates a sort of island where the customer is free to move and go around. Actually the 85% of sales is done in this way. Only in few cases (10%), customers use internet-based shops. Then, the machine is delivered at home and installed by a technician. The customer uses the machine as everybody knows, caring about the loading of the most proper cleaner soaps, the knowledge of the washing programs and its choice, etc.
From Service point of view, Indesit offers only few supporting services such as: warranty, 24 hours assistance and assistance website. However, they are always sold in addiction to product according to a basic Product+Service model and they are really simple, so servitization is limited to the maintenance assistance and spare parts.
The current situation is hereinafter described. After purchasing the product, the customer subscribes a traditional warranty contract of variable duration according to the customer‘s choice (1-3 years). It assures:

- Free maintenance;
- Free spare parts;
- Free delivery at the nearest Assistance Center (when the product cannot be fixed at home);
- Product substitution is it is not reparable;
- On-site intervention of an enterprise technician (only the first 6 months).

The warranty can be also extended (5 years) by a special warranty formula, which extends the basic warranty to 5 years and also offers free on-site intervention, free spare parts also after the first 6 months, and a 10% discount for the purchase of accessories or other aesthetical parts. Furthermore, the customer can register its product on the dedicated website and download the related documentations (warranty conditions, use manuals, etc.). It also offers a special section containing the most common problems and the best solutions, and an on-line shop to directly order spare parts or accessories. Moreover, a free number is 24hours available. Concerning service ideation and development, services are actually conceived and designed after the product in a separated way (product development cycle + service ideation process). Usually first the product comes and then the related services are added to the already existing product by minor changes. It implies that services are defined after the product development. As a consequence, services are ―added‖ to an existing product by minor changes (adding a new component, modifying the SW control to improve some functions, changing the selling strategy, evolving the user interface, etc.). Services are conceived and designed by the marketing staff: it aims to define the solution intended as a service while R&D activities aim to define the ―solution‖ intended as a product. The two flows are organized according to the same 4 stages: Generation, Screening, Exploration and Delivery.

The actual business model of use case is involved the following actors: Indesit, the customer, the sales network (local dealers/retailers/distributors), suppliers, and Universities and research centers. Hereafter their roles are described.

- Indesit is:
  - Product designer and producer;
  - Product seller;
  - Technical Assistance Provider.
- Customer is:
  - Product buyer;
  - Product end-user;
  - Warranty contract subscriber;
  - Person requiring technical assistance.
- Sales network (dealers/retailers/distributors, mainly from large-scale distribution) is:
  - Main product distributor/seller.
- Suppliers are:
  - Technical partners in designing and/or producing some components;
  - External partners providing specific services (molding, electrical boards, etc.).
- Universities (or research centers) are:
  - Scientific or technological partners in developing R&D projects and innovating some specific components or functions.

Actually all actors are spread in Europe (i.e. Universities) and worldwide (i.e. Suppliers, Sales network, Customers) and below figure has represented the enterprise ecosystem explicitly.

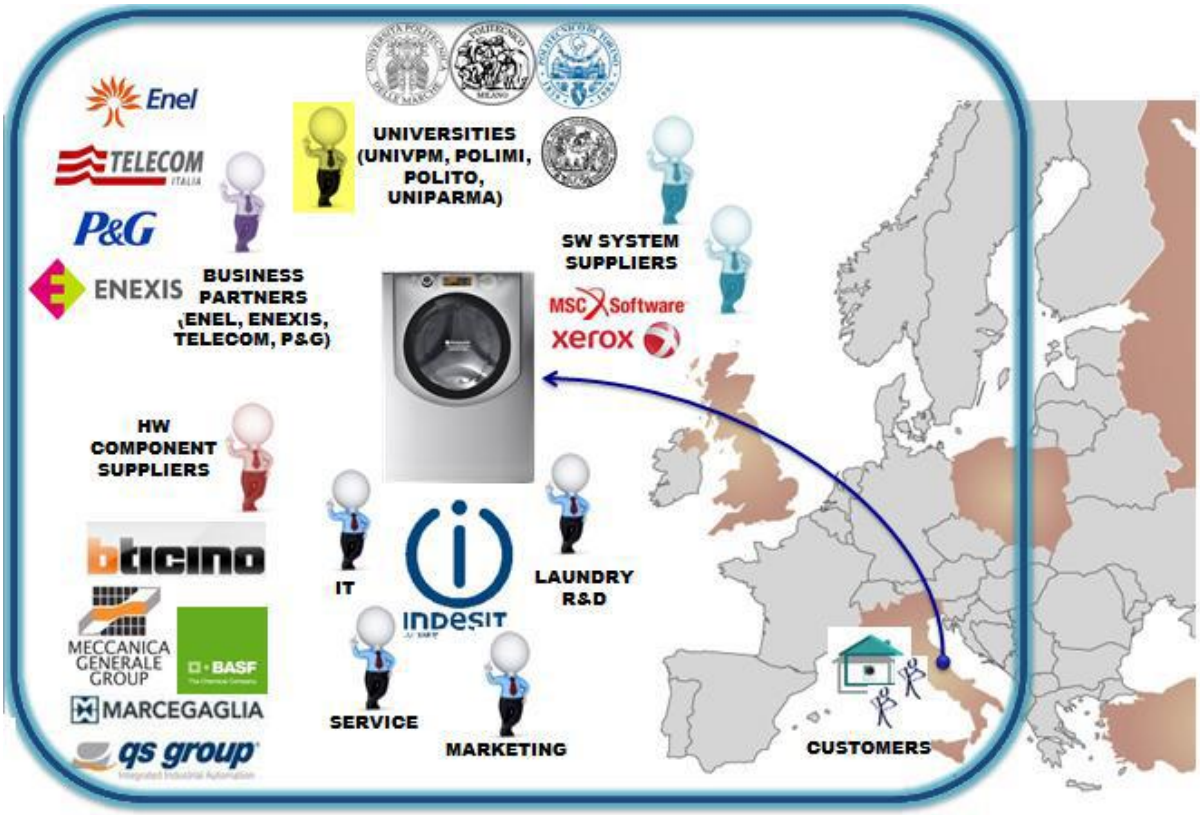


Figure 66 Indesit Ecosystem

According to the servitization process, the validation case actual level is rather low as it is limited to the second level that is selling the physical product. Only few basic services are offered in a traditional way (e.g. warranty, technical support, service call center, etc.). In certain cases the second level of servitization is partially achieved if we consider the maintenance service by warranty contracts and the 24hours assistance service offered by call center and website. Here below the current servitization level of use case has been represented.

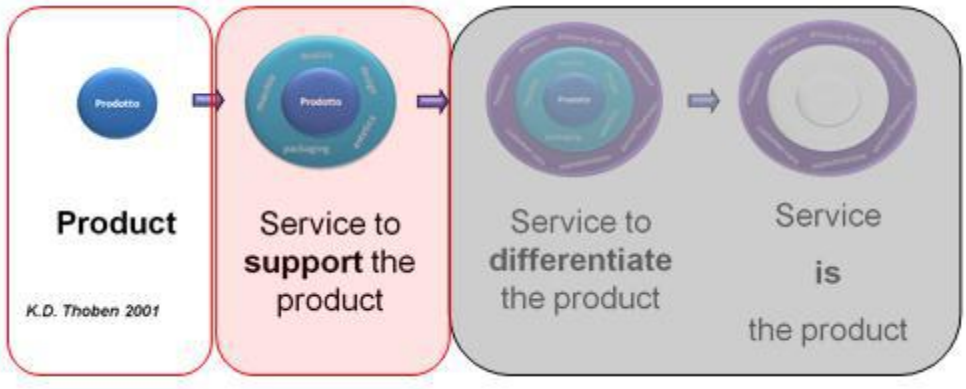


Figure 67 Use case AS-IS Servitization level

In the AS-IS scenario, it is a traditional WM use consisting of common actions (insert clothes, insert the cleaner soap, select the washing program, start the machine, etc.). In the TO-BE scenario the use case can be called “Carefree Washing” because the machine integrates a set of services that make the customer not to care about additional actions (e.g. maintenance, machine control, soap recharge, spare parts, etc.).

5.2 The Product+Service idea: the carefree Washing service (TO-BE Situation)

In order to develop new products and more and more innovative services for its customers, Indesit has been focused on creating a product + service solution starting from the leading products in its portfolio, the washing machine. The washing machine market is an expanded and consolidated sector for Indesit which recently proposed also innovative and advanced solutions aimed at saving energy, smart technology, noise reduction, and self-dosing of detergents. Actually, Indesit still is very product-oriented but wants to increase its service orientation. Indeed, it wants to develop new services that allow providing a carefree washing ecosystem to the customers. For instance with the help of remote control and data services it wants to enable its service business to act preventively and avoid break-downs of the customers´ washing machines. Another example for its new service-focused view is an intelligent soap recharge service. With this service soap shall be automatically delivered to the customers at the time when they need it (soap recharge or refill). Indeed, after having bought a washing machine, the recharge service for soap can be ordered by customers to create additional customer value. Here it could make sense in the future to offer a bundle of product and service. That means Indesit could sell washing machines that have the soap recharge service included for a certain period of time (e.g. during the warranty phase, i.e. the subsequent 24 months after the purchase) and an adapted (i.e. higher) selling price. With the product-related services mentioned above Indesit can support its white goods and differentiate towards competitors. Therefore, the use case has been focused on washing machine; the use case can be defined as "Washing Machine Use". Indeed, among all products, washing machines (WMs) actually represent the greater market share and, as a consequence, also the majority of innovation and research projects have been developed on such an appliance during the last few years. The use case know-how on WMs is wide and robust and a lot of innovative and advanced solutions have been recently applied on it (e.g. energy-saving, high-performance smart technology, silent motion control, auto-dose of soaps and cleaners, etc.). The use case idea started from two considerations: the widespread of WMs inside domestic houses, the worldwide distribution and the underused potential of actual WM electronics. These factors make the WM as the ideal candidate to become a worldwide ecosystem element and to be further developed. Since Use case is still marginally in the Product+Service phase at the present moment, the new scenario has a challenging objective: to move to level 3 of servitization and investigate also the potential of a Product2Service scenario (see figure below). The new service-oriented scenario implies also a change of the current ecosystem, which is composed mainly by internal actors and few external entities involved only in R&D phases. Contrariwise, the TO-BE scenario implies that some external partners will be involved after sales to support new services as service providers. It forces to define a robust and successful business model to make the ecosystem work.

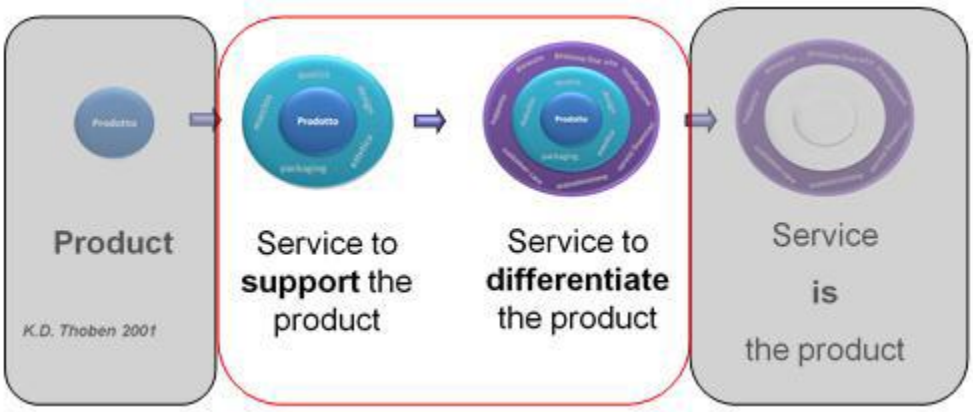


Figure 68 Use Case TO-BE Servitization level

Innovation in the ecosystem is represented by the presence of so many partners and suppliers who must be coordinated in their actions and driven by common rules. It will be complex and challenging. Furthermore, the adoption of an external platform to deliver some services and to analyzed data collected by the machines is a novelty (some systems are usually internal). It implies two contrasting aspects: on one hand, data security and privacy issues must be faced and properly managed; on the other hand, such platform (web-based, shared among numerous partners, etc.) can open new sales channels and can create marketing perspectives. The new service-oriented scenario implies also a change of the current ecosystem, which is composed mainly of internal actors and few external entities involved only in R&D phases. New external partners will be involved to support new services as service providers (i.e. at least an HW-SW component supplier, Utility, Detergent producer). The use case scenario can be summarized by the following figure, providing the overall idea of the main services and the involved actors. The following figure represents the general business scenario where the use case servitization process will take place. The product, the services, the customers and the home network are the main elements. The scenario has been also investigated from the company viewpoint as well as the customer viewpoint.

5.3 Service Functionalities

As mentioned above, Carefree Washing Service offers a set of functionalities to support the customer in washing activities and to realize a ―carefree‖ use of the same product by providing additional services (i.e. machine monitoring, feedback on usage, personalized best practices, tailored marketing offers). In particular, use case scenario is focused on the provision of the following service functionalities:

- WM Monitoring: control of the WM status, global WM data, last cycle data, user habits, by web or mobile applications;
- Best Practice Proposals: provision of personalized feedback and useful advices elaborated on the basis of real user actions and "errors/inefficiencies";
- Marketing Offers: provision of interesting marketing offers elaborated on the basis of the specific user profile and his/her washing habits;
- Detergent supply: provision of personalized detergent offers on the basis of the specific user profile and his/her washing habits, suggestions of ad-hoc WM-related products, and on-line order.

They are to be implemented with the support of the industrial assets. This will affect the three impact categories: Manufacturing, Organization and IT.

Manufacturing impact: the machine needs to be enhanced with further functionalities and components like the zigbee module and the new main board. The zigbee module sends data to a local gateway Connectivity to allow the data passage to the web; the main board reads and stores data thanks to some firmware modifications and upgrading of the setting files.

Organization impact: the ecosystem needs to be properly defined and organized through the partner selection, to choose the best solution, the marketing and R&D collaboration, to realize a feasible product-service offer, and Service Lifecycle Management (SLM) to manage product-service lifecycles.

IT impact: the product architecture requires new technological components and new software applications like data storage from the machine to the web, data elaboration and management to have feedback from customers and delivery platform to deliver services to final users.

5.4 New Ecosystem for TO-BE Situation (VE)

The TO-BE ecosystem is enlarged if compared to the AS-IS ecosystem as new actors are involved. Indeed the ―carefree washing‖ system needs the development of a specific ―carefree washing ecosystem‖ in order to provide the service packages proposed in the previous section. As a consequence, the TO-BE ecosystem involved the actors already involved in the actual situation (Use case, the customer, the sales network, suppliers, Universities) but also some new partners to perform some specific roles inside the new environment. In particular, the customer becomes an active part of the Use case new TO-BE ecosystem and some service providers are involved. Hereafter the actors involved and their roles are described.

- Use case is:
  - Product designer and producer (R&D);
  - Services creator (Marketing);
  - Product+Service seller;
  - Analyst of all data recorded by products and customers (Technical Assistance and Marketing);
  - Technical Assistance Provider;
  - The leader company coordinating all the other partners (service providers).
- Customer is:
  - Product+Service buyer;
  - Product+Services end-user;
  - Warranty contract subscriber;
  - Person requiring technical assistance;
  - Person requiring H&S assistance;
  - Person monitored at home;
  - Person buying cleaner soaps;
  - Person giving feedback on the product and service use (to monitor the Product+Service use and conceive new ad-hoc services)

    - Person co-creating services with use case on the basis of the WM use and implicit/explicit needs.
- Sales network (dealers/retailers/distributors, mainly from large-scale distribution) is:
    - Product+Service distributor/seller.
- Suppliers are:
    - Technical partners in designing and/or producing some components;
    - External partners providing specific services (molding, electrical boards, etc.);
    - Providers of some additional services (maintenance, assistance, etc.
- Universities (or research centers) are:
    - Scientific or technological partners in developing R&D projects and innovating some specific components or functions.
- Mobile application provider (as supplier) is:
    - Developer mobile applications for product remote control and product data monitoring;
    - Provider of the developed mobile applications.
- Cleaners/detergents producer (as partner-supplier) is:
    - Producer of soap cleaners and detergents;
    - Provider of soap cleaners and detergents.
- Local Service delivery provider (as supplier) is:
    - Provider of on-site delivery service about spare parts, soaps, and anything that needs to be delivered at home.
- Disposal and recycling provider (as partner) is:
    - Responsible of product disposal and recycling on-site.
- Health & Safety service provider (as supplier) is:
    - Provider of H&S service and on-site assistance in case of emergency.
- Energy/Water provider (as partner) is:
    - Provider of energy/water facilities to reduce energy/water consumption and costs.

In conclusion, the new ecosystem will be realized with three main actions: Washing machine monitoring (sending information to an external ecosystem), Users monitoring (sending information about end-users actions, e.g. by smart phone), and Service delivery infrastructure (to manage, store and elaborate the system data).

5.5 Scenarios and obtained models

This section will demonstrate the use of the SLMToolBox and the realization of MDSEA in service development. Two different scenarios are discussed and in each scenario different models developed by the SLMToolBox (at BSM and TIM levels) are presented and detailed. The goal behind presenting the developed models is to validate the use of the SLMToolBox as a modeling tool and as being a partial implementation of the MDSEA methodology.

5.5.1 Design a single service for Indesit

This first scenario is based on designing the care free washing service within Indesit i.e. a single enterprise and it describes the phases of the service lifecycle form ideation to execution. Different Models are developed at the BSM and TIM levels.

5.5.1.1 Business Service Model (BSM)

The BSM models developed for this scenario helped to identify the TO-BE process for the production the care free washing service. These models help business experts and Indesit engineers to specify the global procedure and objectives of this service. Extended Actigram Star (EA*) models are used to develop the TO-BE model of the service production process and details the different phases to be applied by Indesit in order to specify, build, and deliver the service.

Figure 71 is the overall service concept of the care free washing service represented with an EA* diagram. The actor responsible for the development of this model is an Indesit service engineer.

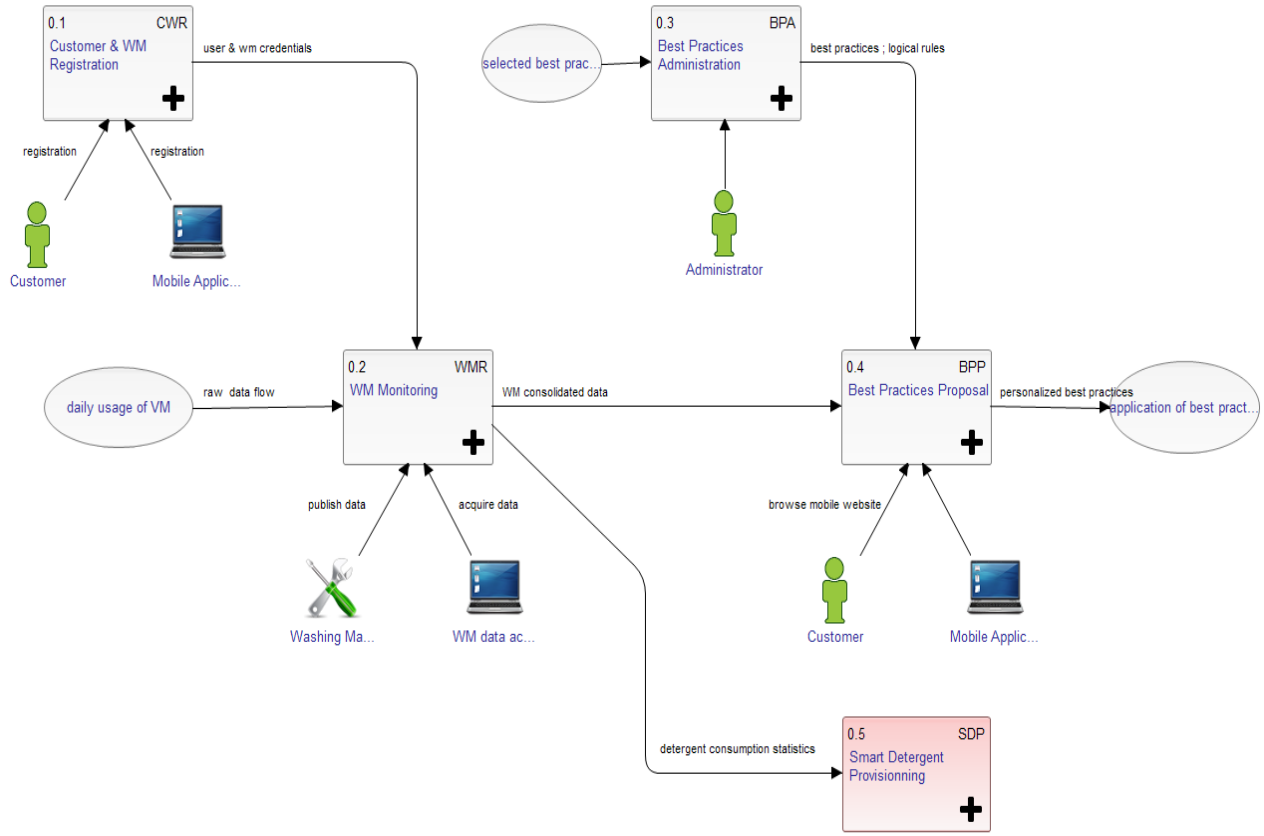


Figure 69 Overall service concept (EA*)

Four main activities constitute the care free washing machine service which is supported by human and IT resources. “Customer and WM registration” is the first activity through which the customer uses the mobile application for registering his washing machine. Credentials of customer and his washing machine are then delivered to the second activity. Daily usage of the washing machine is tracked and used by the “WM Monitoring” to monitor the usage of each registered washing machine and thus keeping track of several information to produce later data and statistics consumed by other activities. WM consolidated data is forwarded to

"best practices proposal" activity, while detergent consumption statistics are delivered to "Smart Detergent Provisioning" activity. "Best Practices Proposal" activity will propose personalized best practices for every registered customer based on data received from their WM. "Smart Detergent Provisioning" receives statistics on detergent consumption and later will deliver detergents to customer based on these statistics.

According to the MDSEA methodology the modeling and designing of the service system starts at the BSM level from a Global point of view. Figure 72 is an EA* diagram representing the global view of the "care free washing service" developed by an Indesit service engineer. The goal is to decompose the service lifecycle management phases.

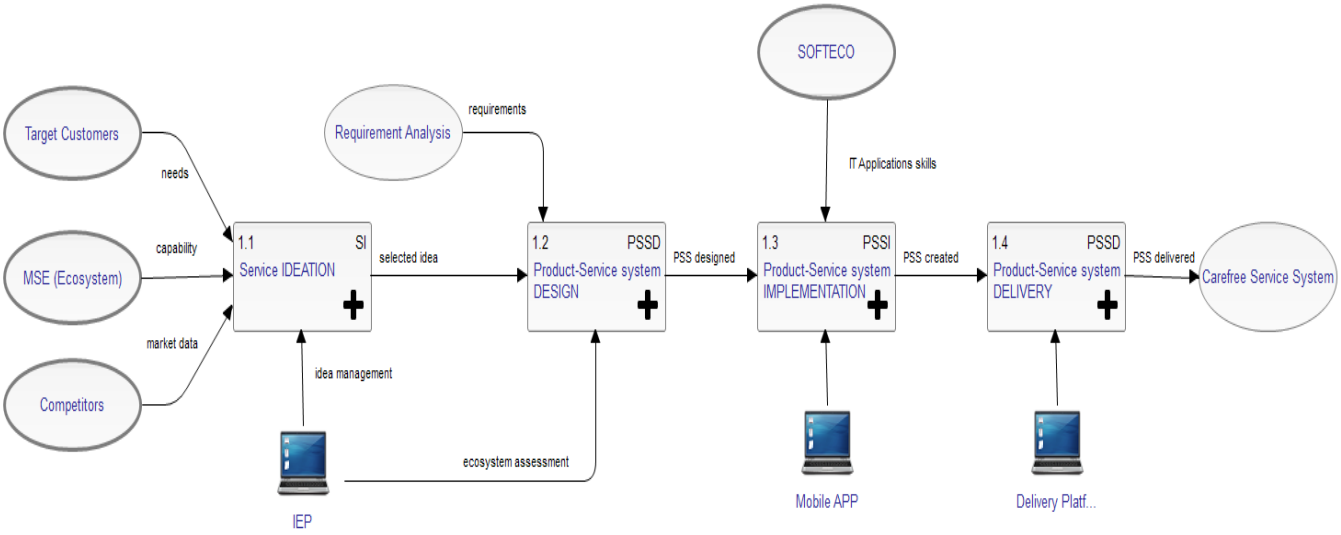


Figure 70 General view (EA*)

Four phases constitute the lifecycle of the "care free washing service". "Service Ideation" activity uses the idea management asset of the Innovation Ecosystem Platform (one of the platforms developed by the MSEE project). It collects information and data using the idea management tool and then the best idea is selected. Next the "Product-Service system design" activity as its name indicates it is responsible for system design based on requirements. The designed system is later to be implemented at the "product service system implementation" activity using IT skills from SOFTECO (technical partner of Indesit) and the Mobile Application Platform (developed by the MSEE project and facilitates the implementation of mobile applications through code generation). After the creation of the mobile application, the "product service system delivery" activity is responsible for the delivery of this service using the MSEE Delivery Platform.

In figure 72 the four phases are represented with structural activities which can be decomposed into other activities and EA* flow elements. Every structural activity will be represented with an EA* diagram. The goal of these diagrams is to communicate the service lifecycle and the different phases needed to produce the service while specifying the resources necessary for the execution of each phase. The diagrams are presented and elaborated in

Annex-4

5.5.1.2 Technology Independent Model (TIM)

At the TIM level UML models are developed by the SLMToolBox and used later by other code generation platform for the generation of care free washing mobile application. The IT models are designed and modeled using UML diagrams and are classified into IT architecture specification and IT software design. The goal of the IT architecture specification models is to describe the high level architecture of the monitoring system and the carefree washing mobile web site. The models are developed by SOFTECO software engineer as being a technical partner to Indesit. Figures 73 and 74 are UML models representing the high level system architecture and the care free washing mobile website respectively.

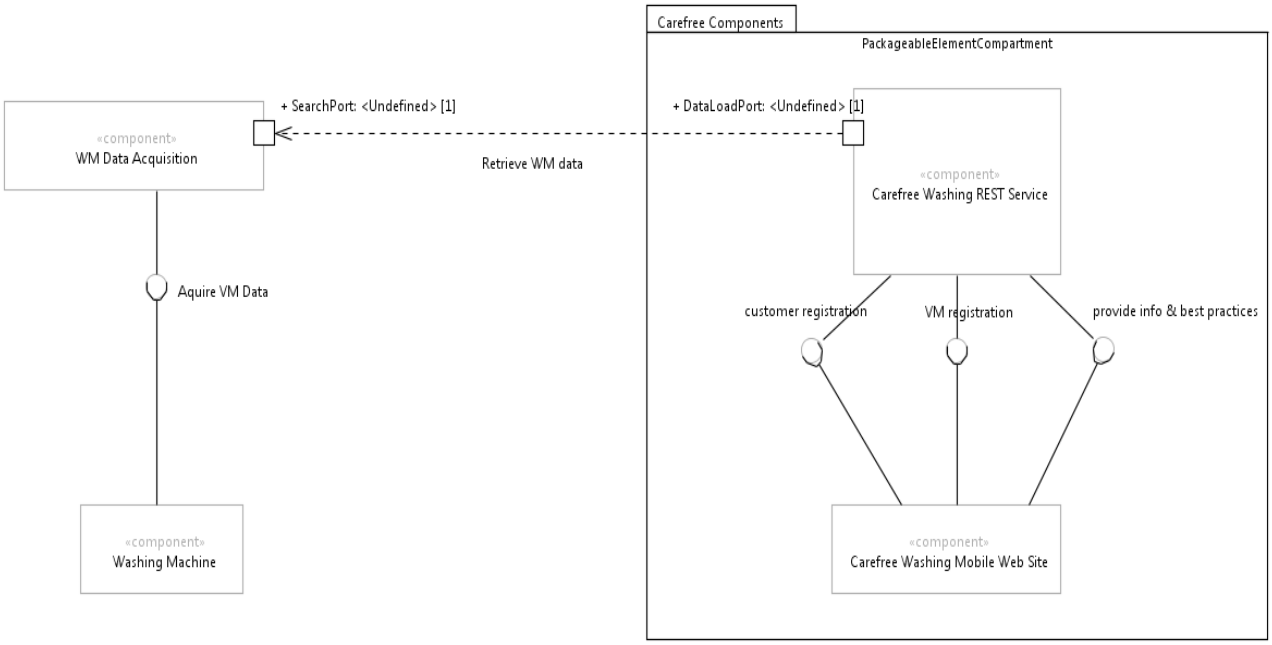


Figure 71 High level system architecture (UML)

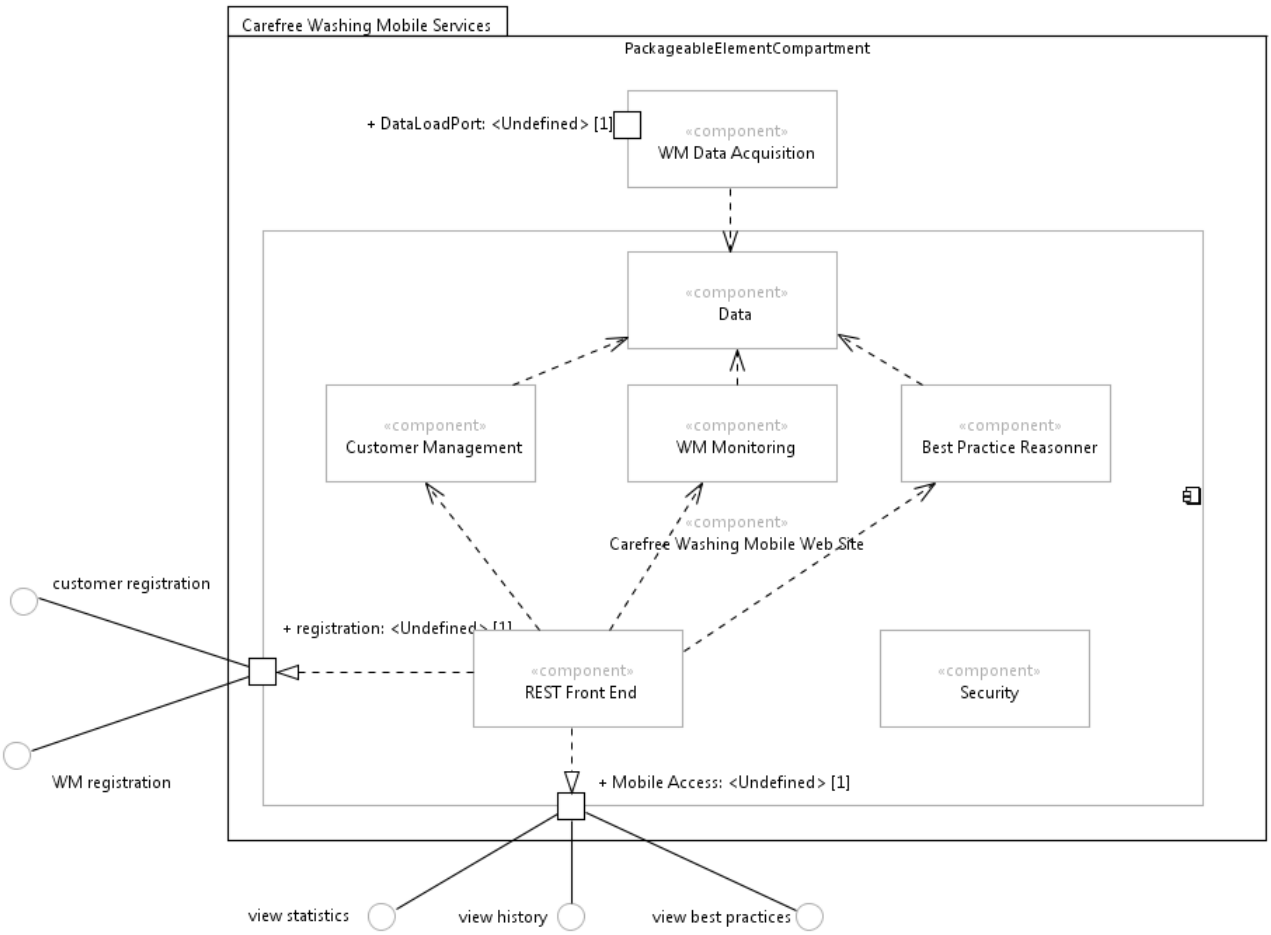


Figure 72 Care free washing machine website (UML)

The IT software design is represented by a UML class diagram in order to build the data (object model) to be manipulated by the mobile website. Figure 75 represents the Washing Machine usage data.

The developed IT models at TIM level needs to be implemented at the TSM level. The SLMToolBox doesn't support implementation and code generation since it only covers the BSM and TIM abstraction level. As a result, all models are exported and pushed into the MSEE model repository. Software developers will later import these models into the mobile development platform which will generate the code necessary for the development of care free washing mobile website.

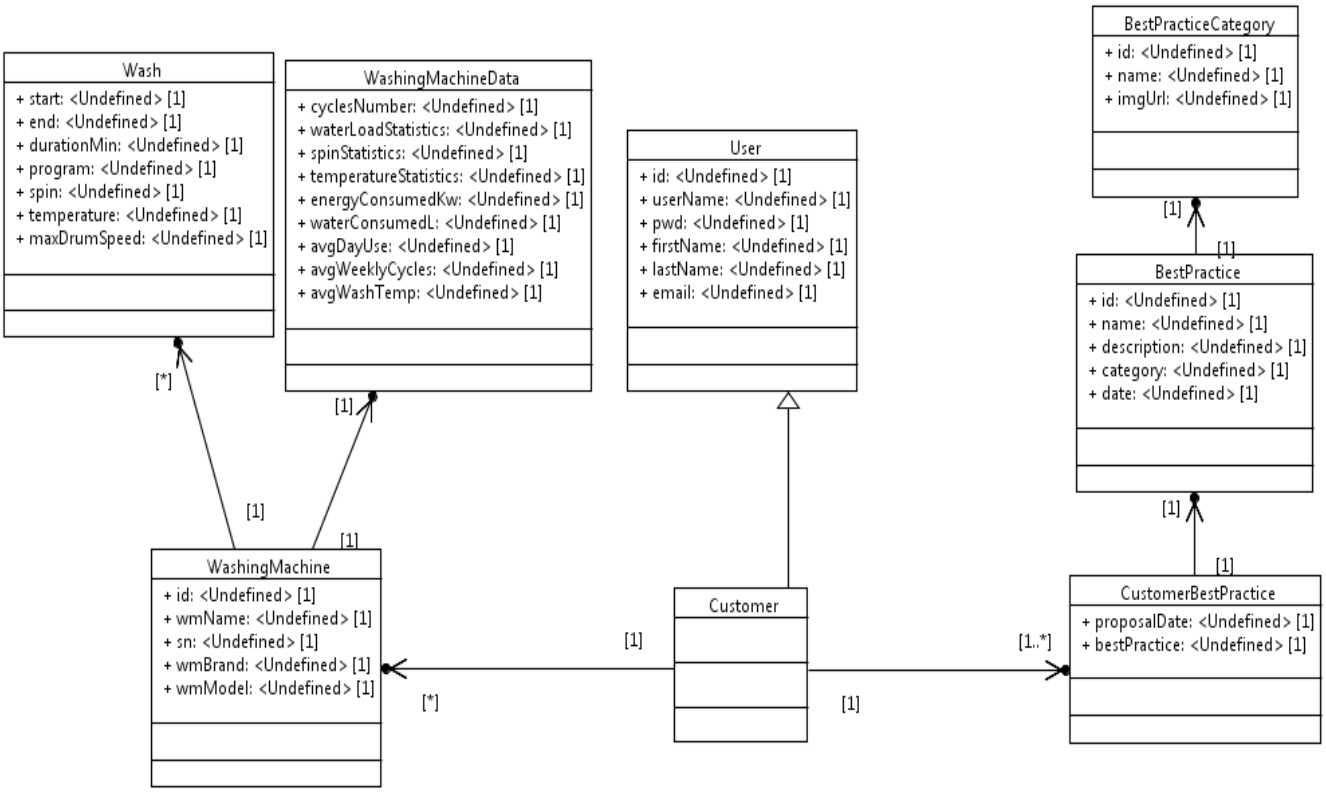


Figure 73 WM usage data (UML)

5.5.2 Design a composite service within the Indesit VME

This scenario demonstrated the development of the "Smart Detergent Provisioning" composite service models within the Indesit VME defined earlier in section 5.5.4. Models in this scenario reveal the use of SLMToolBox in modeling collaborative models and diagrams between partners in the VME. In addition, it demonstrates how model transformation is applied to transform EA* models into BPMN models.

5.5.2.1 Business Service Model (BSM)

In figure 71 the activity "Smart Detergent Provisioning" is regarded as an offer enriching the overall service. It is a composite service in which several partners are involved in the realization process. Figure 76 describes the collaboration within the VME for the TO-BE "Smart Detergent Provisioning" offer (service).

From the diagram we notice two VME partners (detergent supplier and B2C supply partner) involved with Indesit in this service. Depending on statistics received from the "WM Monitoring" activity and the analysis of these data, a detergent order is generated to the B2C supply partner. In figure 76 two external assets are represented in this diagram (this is indicated by the black circle with an arrow inside):

- B2C Onsite Delivery: an external service provided by the B2C supply partner
- Ecofriendly detergent: a resource (external to Indesit) referencing an asset provided by the detergent supplier partner.

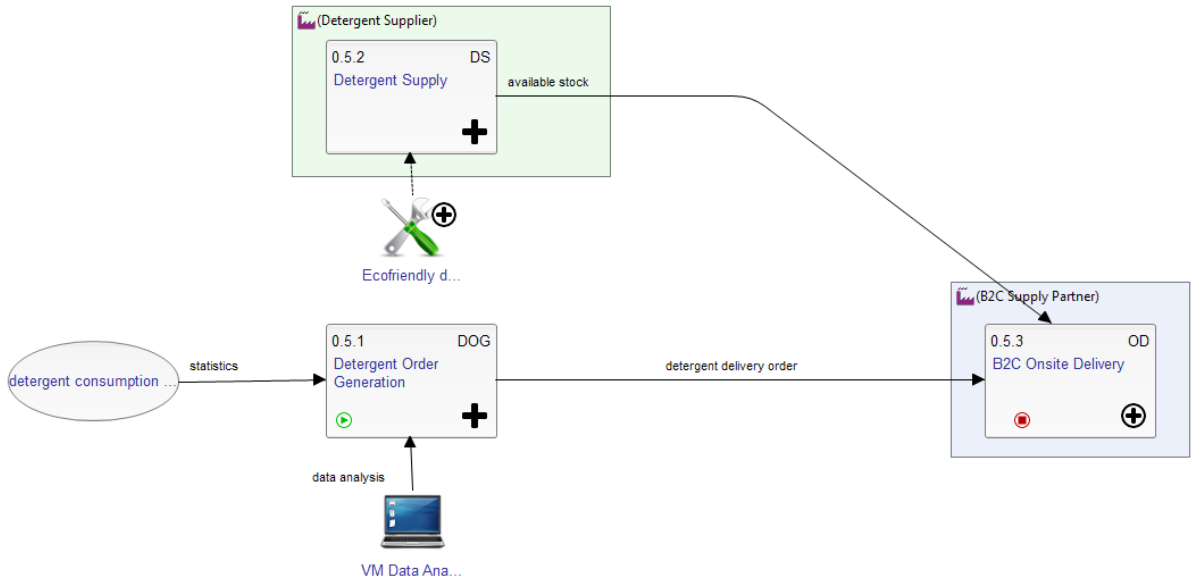


Figure 74 Smart Detergent Provisioning (EA*)

In fact, the SLMToolBox through its EA* graphical editor is capable of connecting resources to VME tangible and intangible assets. Every partner of the VME, who is willing to share one of its resources with other partners, registers the desired resource in an asset repository and provides specific related information. In this case other partners can browse the assets repository and choose external assets (tangible or intangible) to be used in their service production. In addition, activities can be also connected to external services proposed and registered by other partners in a service repository. In an Indesit EA* process diagram, the representation of tangible/intangible assets and external services designs the collaboration between VME partners in the production of the required service.

Figure 77 is a GraiGrid diagram with a goal to model the governance of the VME (functions, levels, inter-relationships). Three VME partners (B2C Supply partner, Detergent Supplier, and Indesit) are presented in the diagram. For every partner, one or several departments are designed with each department possessing several decision centers. These decision centers are divided over three decision levels: strategic, tactical, and operational. In addition, Communication, sequence, and information flows between decision and flow centers are modeled.

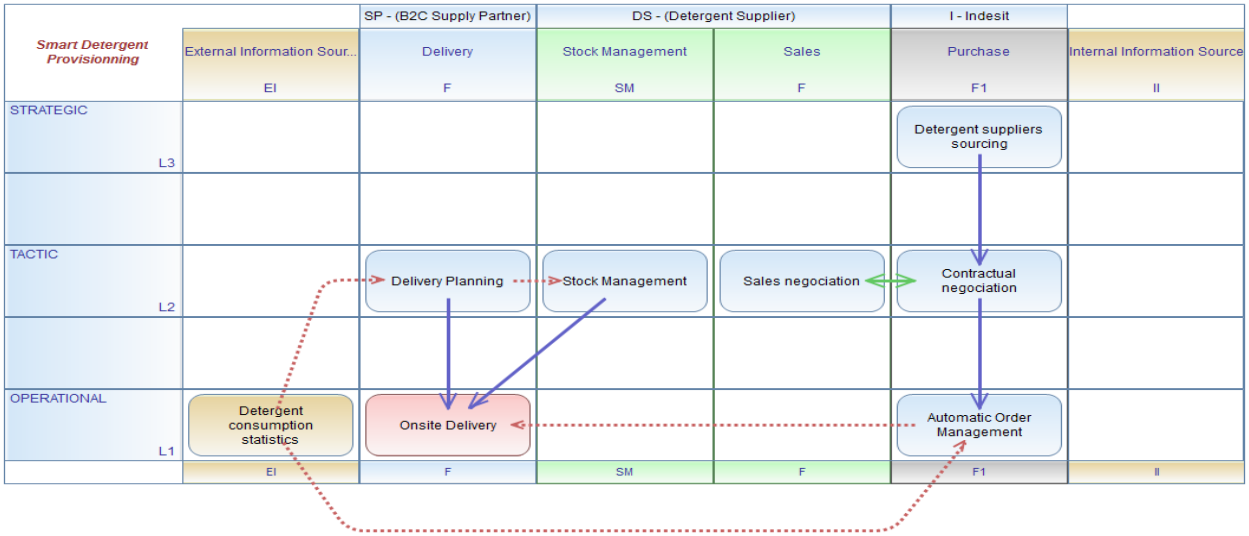


Figure 75 Smart Detergent Provisioning (GraiGrid)

With the GraiGrid editor presented in section 5.3.2.2, we can model the decision and strategic view of the service to be produced in collaboration between VME partners. Involved partners, responsible organizational structures, decision and information centers, and the flow of information are all modeled and visible in the diagram. In addition, the user can define objectives, decision variables, and performance indicators that are associated to a specific decision centers. Figures 78, 79, and 80 reveals how the user can define these terms via the GraiGrid editor. The user starts by selecting a decision center (figure 78) and clicking the PI green icon, a wizard is opened with three available tabs (objectives, decision variables, and performance indicators). The wizard serves in adding, editing, and deleting objectives, decision variables, and performance indicators (figure79). In addition, performance indicators can be imported form a list proposed by the scientific research work of several MSEE partners (figure 80).

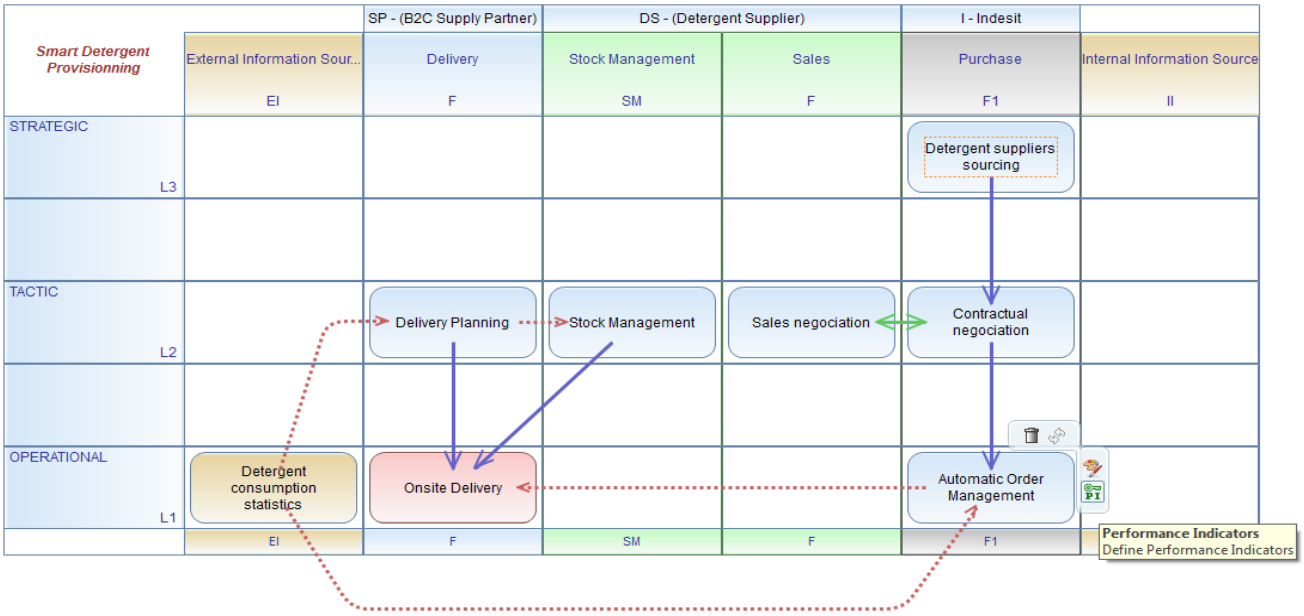


Figure 76 Performance indicators (1)

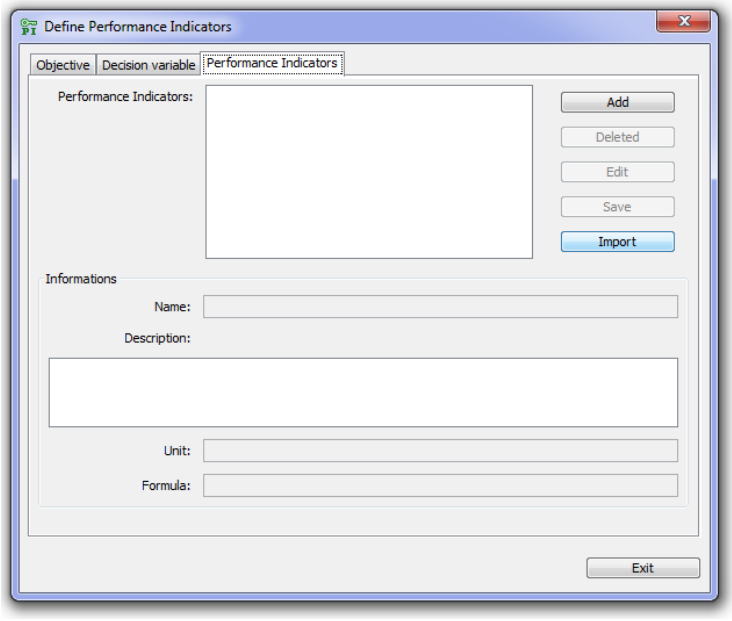


Figure 77 Performance indicators (2)

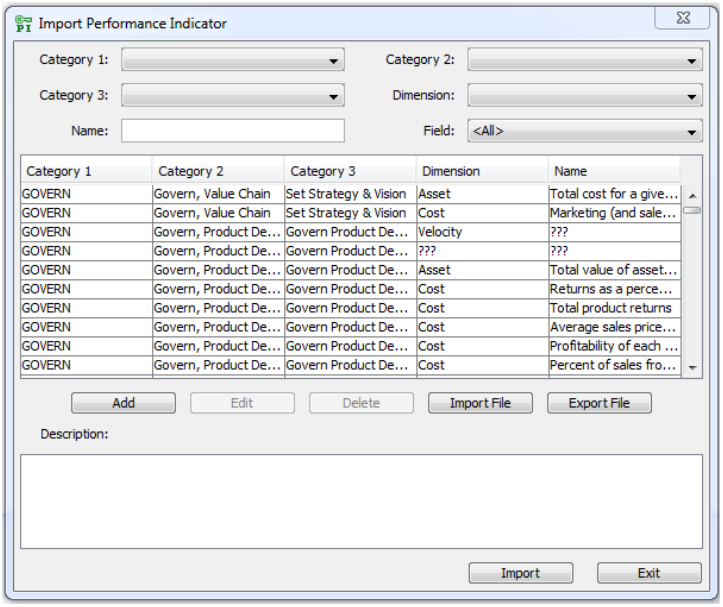


Figure 78 Performance indicators (3)

5.5.2.2 Technology Independent Model (TIM)

In previous sections, we introduced the different features provided by the SLMToolBox. One of these features was the model transformation from EA* models to BPMN models. This specific transformation will help different actors to communicate at different abstraction levels and reuse pre-developed models as explained earlier. Using the SLMToolBox we transformed the "Smart Detergent Provisioning" EA* diagram (figure 76) into a BPMN diagram. Figure 81 is the result obtained from the transformation using the SLMToolBox. Graphical objects are not well arranged due to the lack of auto positioning in the BPMN modeler integrated in the SLMToolBox. The user is then invited to rearrange the graphical objects to obtain a user friendly diagram (figure 82). The obtained diagram can be regarded to be at the top TIM level and needs to be enriched by a software engineer. In figure 83, the software engineer specifies the logical rules and messages which must be handled by the automated part of the supply process.

After the enrichment of the Smart Detergent Provisioning BPMN diagram at TIM Level and using the SLMToolBox, the diagram is exported and stored in MSE model repository (connection to the model repository can be accomplished automatically using the SLMtoolBox). Software engineers using the MSEE development platform will import the BPMN diagram (TSM level) and enrich it with technical information before being published into the Innovation Ecosystem Platform (IEP). The IEP will later execute the BPMN diagram to orchestrate the process's execution after implementing the necessary web services.

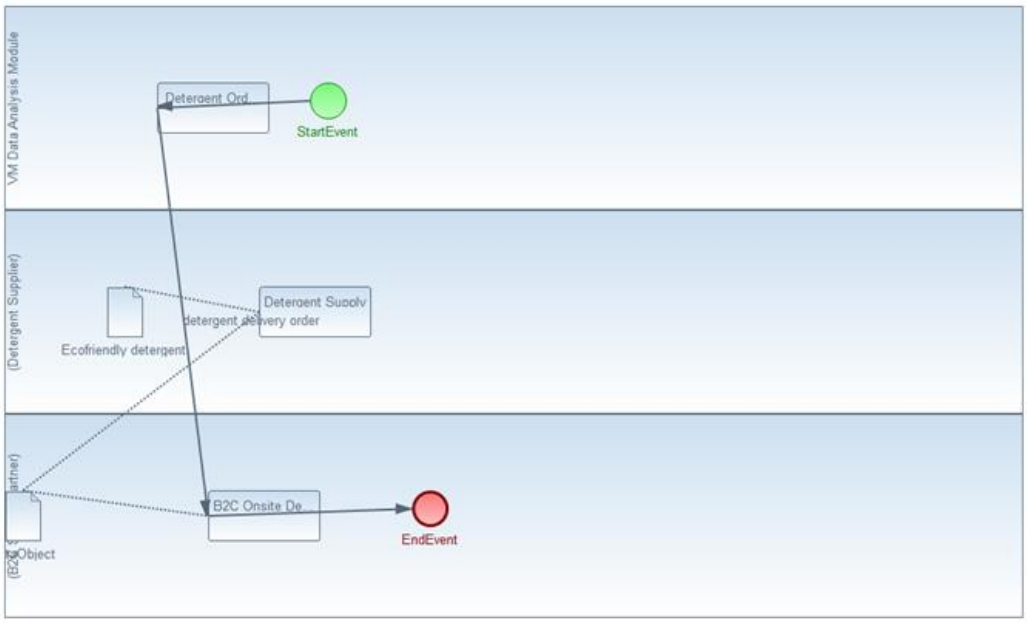


Figure 79 RAW Smart Detergent Provisioning (BPMN)

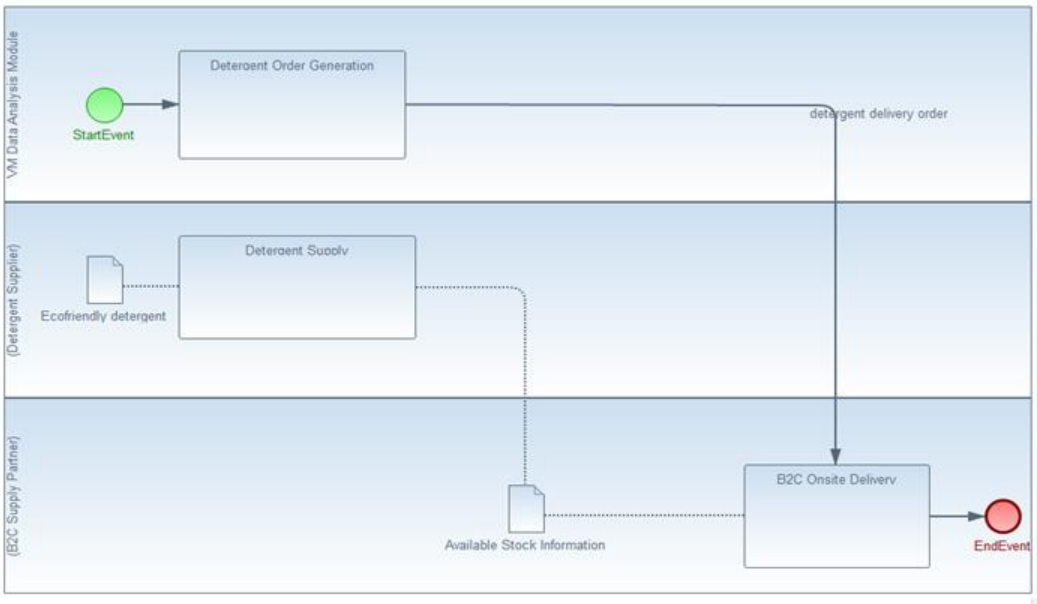


Figure 80 Rearranged Smart Detergent Provisioning (BPMN)

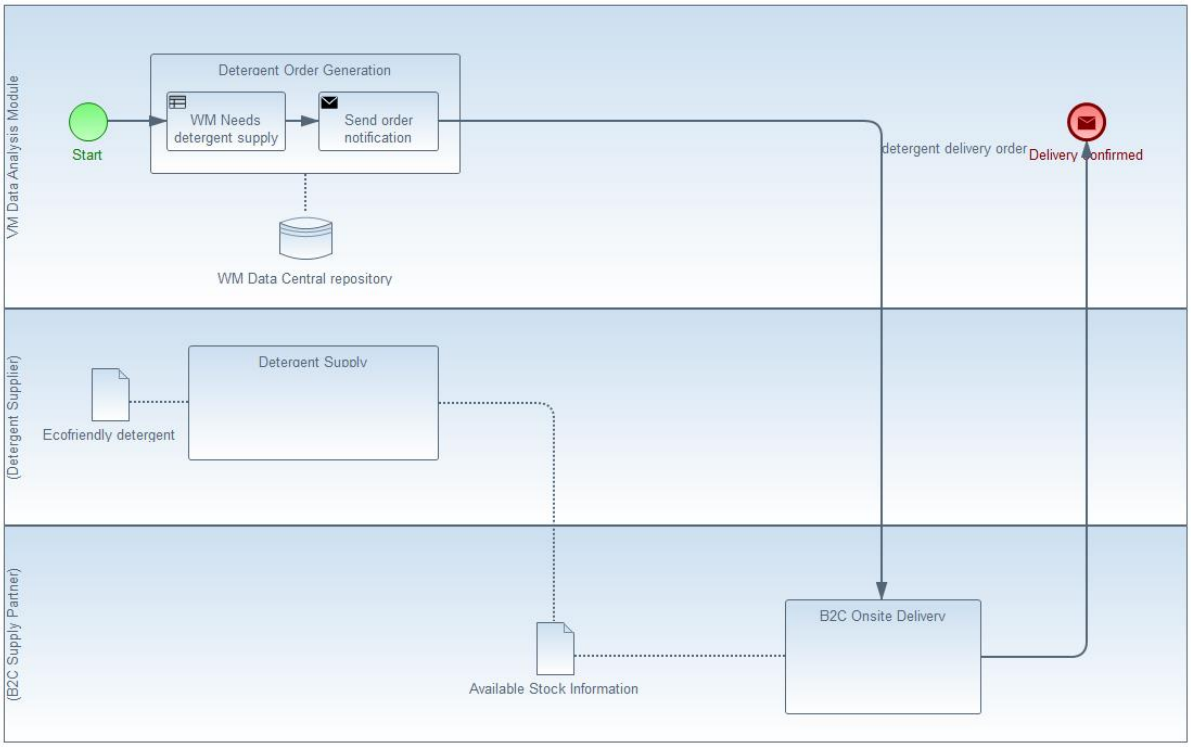


Figure 81 Enriched Smart Detergent Provisioning (BPMN)

5.6 Conclusion on the Indesit Use Case

In section 5.5, we presented the Indesit use case as being one of the MSEE manufacturing partners. The AS-IS and TO-BE situations were introduced while explaining the new collaborative service to be produced by the Indesit Virtual Enterprise (VE). The shift from product to product+service systems is accompanied by the use of the MSEE project's results. MDSEA concepts and the use of SLMToolBox were investigated during the servitization process of Indesit. Several models were produced at the BSM and TIM levels and then delivered to other MSEE platforms in order to proceed with the implementation of the service.

At BSM level GraiGrid models were developed to model the strategic and tactical views of the service. In addition objectives and performance indicators were imported into the GraiGrid models for governance and monitoring reasons. EA* diagrams were developed through different scenarios with diagrams targeting collaboration processes between different partners while others represented the service production lifecycle phases to be held by Indesit and its partners and thus can be used for communication between partners and a representation of the service production in the TO-BE VE. In certain diagrams external assets and services were referenced using the EA* editors that can connect to assets and service repository.

At TIM level UML models were developed for the Care Free Washing mobile application. These models were exported to the MSEE mobile platform and used for the automatic creation of the application's source code. In addition, the transformation from EA* models to BPMN models were presented. BPMN diagram was created form existing EA* diagram using the model transformation feature of the SLMToolBox, then the diagram was rearranged and enriched with IT information and details. The enriched diagram was finally exported to the model repository and used by other platform for execution.

6. Conclusion

Chapter 5 presented the SLMToolBox as a modeling and simulation tool developed during this thesis work. We provided a system and technical overview, and explained the implementation of MDSEA and simulation principles inside this tool. Also, the different features constituting the tool are presented and detailed. At the end of the chapter, the Indesit case study is illustrated with a description of current situation and position in the servitization process, the services to be introduced to their product, defined scenarios, and obtained results.

After presenting our research work and contributions, the following chapter will present future perspectives for our work and produced assets.

General Conclusion and Perspectives

The previous chapters represented the contribution of our thesis, this chapter will conclude these results and highlights future perspectives.

1. General Conclusion

During this thesis we contributed to the development of the Model Driven Service Engineering Architecture (MDSEA) which target was helping manufacturers in their business transformation shift towards servitization. We specified a process modeling language, the Extended Actigram Star (EA*) language with abstract and graphical syntax identified. An EA* metamodel is developed which takes into consideration conceptual and implementation concerns. The metamodel permits the development of graphical editors used for the creation of graphical diagrams. In addition, model transformation and its utility in MDSEA were discussed, in particular our proposition of transforming EA* to BPMN models and BPMN to DEVS models. Both model transformations were based on previous research work and it answered challenges met by researchers during their work. These model transformations were executed through different steps: defining the mapping of concepts between both source and target models, implementation of the identified mapping using the Atlas Transformation Language (ATL), executing the ATL transformation, implementation of an XSLT style sheet used for the creation of a diagram and graphical objects, and finally executing the XSLT style sheet to add graphical objects to the model. We also presented a DEVS editor and simulator which was used to simulate business processes using the DEVS formalism. The simulator provides a simulation report and animation features. At the end we introduced the Service Lifecycle Management Tool Box (SLMToolBox), a modeling and simulation tool we participated in its development at Hardis Group. This tool is a partial implementation of the MDSEA methodology and in which we integrated an EA* editor, the model transformations specified, and the DEVS editor and simulator.

Chapter 1 was the general introduction of the thesis, it reported the context in which the thesis was involved and the problem trying to participate in solving. The context in general is about servitization and how manufacturers are shifting their business in a transformation process towards services. On the other hand, the problem is the lack of methods, methodologies and tools to accompany those manufacturers in their transformation. MSEE project tried to answer and contribute to these challenges and the thesis being part of the MSEE project has contributed in the development of specific solutions. We then presented the organization of the thesis.

Chapter 2 was the state of the art of this thesis, it presented various work related to our domain of research and study. It introduced enterprise modeling and a group of known methods (CIMOSA, GIM, and ARIS), enterprise interoperability and a list of several approaches and frameworks (IDEAS, LISI, and ATHENA), Model driven development and in particular MDA and MDI, three modeling languages directly related to our work (GRAI Extended Actigram, BPMN, and DEVS), and finally a list of business process and DEVS based simulation tools.

Chapter 3 was our first contribution; it introduced the Model Driven Service Engineering Architecture (MDSEA). The problem statement that triggered the development of MDSEA, is well detailed, starting from the emergence of servitization and the shift from product systems to service systems, then to service systems and service lifecycle management and finishing the problem statement with service system's modeling. Then we introduced MDSEA as a contribution of our research work, detailing its three abstraction levels and the proposed

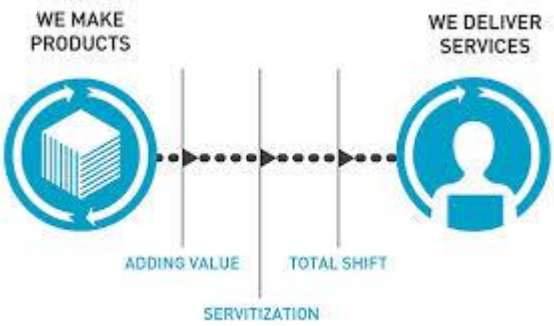
modeling languages at each level. In addition, we proposed a process modeling language Extended Actigram Star (EA*) which we developed based on GRAI Extended Actigram. In order to better explain EA*, we mentioned its scope, overview of the language, its abstract syntax, graphical representations, and its connectivity constraint. At the end of this chapter, we presented the model transformation from EA* to BPMN which we developed and implemented as being part of the MDSEA methodology. Also an example of this model transformation is shown.

Chapter 4 was dedicated to simulation of business processes in MDSEA. It started presenting the problem we are trying to answer by simulation and in particular by our developed DEVS simulator. We then justified the choice of DEVS formalism through its basic characteristics. The second model transformation BPMN to DEVS models which we have developed was proposed in addition to the DEVS simulation implemented in the SLMToolBox (execution, results, and animation). And finally an example of the transformation of BPMN models to DEVS and the simulation of obtained model was demonstrated.

Chapter 5 is the final chapter containing the thesis's contributions. It presented the Service Lifecycle Management Tool Box including system's overview, technical overview, the implementation of MDSEA and simulation in the SLMToolBox. A use case study is detailed at the end of this chapter in order to validate the obtained results.

Figure 84 summarizes the context, problem and contribution of the MSEE project and this thesis in particular. In the context, manufacturers started shifting their production towards services (servitization). Then these manufacturers agreed to collaborate between them in order to compete in the market and produce services. This collaboration leads to the formation of ecosystems and virtual manufacturing enterprises. The transformation is governed through the use of MDSEA, SLMToolBox, Simulation and other MSEE methods and platforms.

Context & Problem



Contribution

MDSEA

SLMToolBox

Simulation

Other MSEE methods and platforms

Figure 82 context, problem, and contributions

2. Perspectives

The MSEE project members and with the effort of I-VLab have worked on a proposal for standardization of Service Modeling Language (SLM) where MDSEA acts as a framework for the proposed language. The introduction of the proposal was as follows *"There is no language standard in ISO or CEN for the modelling of service system. Some existing service modelling languages mainly focus on IT related service or Web service. Most of existing enterprise modelling languages are relevant to service in VME and can be reused to model part of a service system in the context of VME. In order to cover the whole modelling requirements for service system engineering, the concepts of those modelling languages need to be integrated and mapped one to another. A standardized Service Modelling Language (SML) and its associated meta-model is seen as an important issue to avoid hazardous and fragmented development in this domain. A Model Driven Service Engineering Architecture (MDSEA) adapted from MDA/MDI acts as a framework for the proposed service modelling language…"* Several meetings had been held for this purpose at Brussels and an action plan was initialized for the various actors. If the proposal is accepted, more future contribution by various partners will be needed to produce a well-structured and specified language.

Some European projects in ICT achieve good results, particularly in the development of methodology supported by software. Unfortunately, sometimes the dissemination and the development of these software stop after the end of the project. For this reason, different partners and actors (exterior to the MSEE project) have met several times to form a community group around the SLMToolBox. The members of the group will take the engagement and to facilitate further development of the SLMToolBox based on MDSEA. The main interests for the partners to join the community group could be to:

- Extend the Method and/or the Model (research interest)
- Deliver, maintain, support and extend the Tool (software) (development interest)
- Disseminate, exploit and adapt the asset for further experimentation and use (innovation interest)

Several projects and universities have continued the development of the SLMToolBox to meet its requirements. The NOCIFEL project as an example which is a French project whose goal is to develop an innovative and modular platform for managing good's transportation has customized the SLMToolBox's Extended Actigram Star editor to meet transportation requirements. In addition, [UNINOVA] an independent and nonprofit research institute in Lisbon and MSEE partner is willing to adopt and extend the SLMToolBox in some of its projects. Besides, the University of Bordeaux is studying the possibility to use the SLMToolBox for DEVS modeling and simulation courses.

References

Agostinho C., Sarraipa J., Gonçalves D., Jardim-Goncalves R. Tuple-based semantic and structural mapping for a sustainable interoperability. DOCEIS'11. Costa de Caparica, 2011.

Amyot D., Farah H., Roy J. F. Evaluation of Development Tools for Domain-Specific Modelling Languages. System Analysis and Modelling: Language Profiles, Vol. 4320, 2006, pp 183-197

Andrews T. et al. Business Process Execution Language for Web Services. Version 1.1. 2003 available at: http://xml.coverpages.org/BPELv11-May052003Final.pdf

ATHENA Integrated Project. Framework for the Establishment and Management Methodology, Deliverable DA1.4, 2005.

ATHENA, Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications, FP6-2002-IST1, Integrated Project, 2003.

ATL official site: http://www.eclipse.org/atl/.

Baines T.S., Lightfoot H.W., Benedettini O., Kay J.M. The servitization of manufacturing: A review of literature and reflection on future challenges. Journal of Manufacturing Technology Management, Vol. 20(5), 2009, pp. 547-567.

Baines T., Lightfoot H., Evans S., Neely A., Greenough R., Wilson H. State of the art in product-service systems. Journal of Engineering Manufacture Part B, 2007, pp.1543–1551.

Banks J., Carson J. S., Nelson B. L., Nicol D. Discrete-event system simulation. 4th ed. Englewood Cliffs, Prentice-Hall, 2005.

Behrend S., Jasch C., Kortmap J., Hrauda G., Firzner R., Velte D. Eco-service development. Reinventing supply and demand in the European Union. Greenleaf Publishing Ltd., 2003.

Bell M. Service-Oriented Modelling: Service Analysis, Design, and Architecture. Wiley, 2008.

Boughnim N., Yannou B. Using Blueprinting Method For Developing Product-Service Systems. International conference of Engineering Design (ICED), 2005, Melbourne, Australia.

Bourey J. P., Grangel R., Doumeingts G, Berre A. Deliverable DTG 2.3 - Report On Model Driven Interoperability. InterOP, 2007 (report available at: http://interopvlab.eu/ei_public_deliverables/interop-noe-deliverables/tg2-model-driven

Bourey J. P., Grangel S. R., Doumeingts G., Berre A. J. Report on Model Driven Interoperability. Deliverable DTG 2.3, INTEROP NoE, April 2007, pp. 91. Available from http://www.interop-vlab.eu/ [accessed 15 June 2009]

BPMN2 Modeler official site: http://eclipse.org/bpmn2-modeler/

Brezet, Bijma, Ehrenfeld, Silvester. The design of eco-efficient services; Method, tools and review of the case study based 'Designing Eco-efficient Services' project. Delft University of Technology, Design for Sustainability program, 2001.

C4ISR, Architecture Working Group (AWG), Levels of Information Systems Interoperability (LISI), March 30, 1998.

Cardoso J., Pedrinaci C., Leidig T., Rupino P., De Leenheer P. Open semantic service networks. Exploring Services Science, Lecture Notes in Business Information Processing, Vol. 143, 2013, pp 141-154.

Carosi A., Heydari M., Zanetti C., Taisch M., Ducq Y. Service Performance Assessment: A PI Toolset Methodology for VEs. In Proceedings of IFIP WG 5.7 International Conference, APMS 2014, Ajaccio, France, pp 691-698, 2014,

Çetinkaya D., Verbraeck A., Seck M. D. Model Transformation from BPMN to DEVS in the MDD4MS Framework. In Proceedings of the 2012 Symposium on Theory of Modeling and Simulation, article No. 28, 2012.

Charalabidis Y., Gionis G., Moritz Hermann K., Martinez C. Enterprise Interoperability Research Roadmap, Draft Version 5.0, 2008. Available from ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/enet/ei-roadmap-5-0-draft_en.pdf [accessed 20 December 2010]

Chen D., Doumeingts G. The GRAI-GIM reference model, architecture and methodology. Architectures for Enterprise Integration, IFIP Advances in Information and Communication Technology, 1996, pp 102-126.

Chen D., Vallespir B., Doumeingts G. GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology. Computers in Industry, 33(2-3), 1997.

Chen D., Vernadat F. Enterprise interoperability: a standardisation view, in: K. Kosanke, et al. (Eds.), Enterprise Inter-and-Intra Organisational Integration, Kluwer Academic Publishers, 2002, pp. 273–282 (ISBN 1-4020-7277-5).

Chen D., Vernadat F. Standards on enterprise integration and engineering - a state of the art, International Journal of Computer Integrated Manufacturing 17 (3), 2004, pp. 235-253.

Chen D. Framework for enterprise interoperability. In Proceedings of the Workshops and the Doctorial Symposium of the Second IFAC/IFIP I-ESA International Conference: EI2N, WSI, IS-TSPQ, pp.77-88, 2006.

Chesbrough H., Spohrer J. A Research Manifesto for Services Science. Communications of the ACM - Services science, 2006, Vol. 49 Issue 7, pp. 35-40.

Clark T., Jones R. Organizational Interoperability Maturity Model for C2. Department of Defense, Canberra, Australia, 1999.

Czarnecki K., Helsen, S. Feature-based survey of model transformation approaches. In IBM Systems Journal, Model-driven software development, 2006, Vol. 45 Issue 3, pp. 621-645.

Cetinkaya D., Verbraeck A., Seck M. D. Model transformation from BPMN to DEVS in the MDD4MS framework. In Proceedings of the 2012 Symposium on Theory of Modelling and Simulation - DEVS Integrative M&S Symposium, p.1-6, Orlando, Florida, 2012

Doumeingts G., Ducq Y. Enterprise Modelling techniques to improve efficiency of enterprises. In International Journal of Production Planning and Control - Taylor & Francis, 2001, Vol. 12(2), pp. 146-163

Doumeingts G., Vallespir B., Chen D. GRAI grid decisional modelling. In International Handbook on Architecture of Information System, Springer Verlag, 1998, pp. 313-337.

Eclipse official site: http://www.eclipse.org/.

EEF official site: http://wiki.eclipse.org/EEF.

EFFRA (European Factories Of the Future Research Association), FoF (Factories of the Future). Multi-annual roadmap for the conceptual PPP under Horizon 2020. 2013. Pdf can be downloaded from http://bookshop.europa.eu/fr/factories-of-the-future-pbKI0213266/

Elvesæter B., Hahn A., Berre A., Neple T. Towards an Interoperability Framework for Model-Driven Development of Software Systems. Interoperability of Enterprise Software and Applications, 2007, pp. 409-420.

EMF official site: http://www.eclipse.org/modelling/emf/.

ENSEMBLE. Deliverable 2.1 EISB State of Play Report version 1.00. 2007. Available from http://www.fines-cluster.eu/fines/wp/d21/ [accessed 07 July 2011].

FlnES Future Internet Enterprise Systems (FlnES) Cluster. Cluster Book ICT2010 EVENT VERSION- June 2010 http://cordis.europa.eu/fp7/ict/enet/documents

FP7 – FoF-ICT-2011.7.3. Manufacturing SErvice Ecosystem Project. 2011. http://www.msee-ip.eu/

Furrer O. Le rôle stratégique des services autour des produits. Revue Française de Gestion, 1997, n° 113, pp. 98-108.

Garredu S., Vittori E., Santucci J.F., Bisgambiglia P. A Meta-Model for DEVS - Designed following Model Driven Engineering Specifications. In Proceedings of SIMULTECH, 2012, pp. 152-157.

Gebauer H., Friedli T. Behavioural implications of the transition process from products to services. Journal of Business & Industrial Marketing, 2005, Vol. 20, No. 2, pp. 70-80.

Gebauer H., Friedli T., Fleisch E. Success factors for achieving high service revenues in manufacturing companies. Benchmarking: An International Journal, 2006, Vol. 13, No. 3, pp. 374-86.

Gonçalves R., Agostinho C., Garção A. A reference model for sustainable interoperability in networked enterprises: towards the foundation of EI science base. Computer Integrated Manufacturing, 2012, 25(10), pp. 855-873.

Gorka B., Larrucea X., Elvesæter B., Neple T., Beardsmore A., Friess M. A Platform Independent Model for Service Oriented Architectures. Enterprise Interoperability, Ed, 2007, London, S., pp. 23-32.

Grangel S. R., Cutting-Decelle A. F., Bourey J.P. A UML profile for transforming GRAI Extended Actigrams into UML. In Proceedings of the 17th IFAC World Congress, Session: Architectures and Software Tools for Enterprise Integration and Networking in Manufacturing, Paper ThB24.3, Séoul, Korea, 2008.

Graphiti official web site: http://www.eclipse.org/graphiti/.

Gummesson E. Exit Services Marketing- Enter Service Marketing. The Journal of Customer Behaviour, 2007, 6(2), 113-141

Hamri M., Zacharewicz G. Automatic generation of object-oriented code from DEVS graphical specifications. In Proceedings of WSC '12 Winter Simulation Conference, 2012, Berlin, Germany, Article 409.

IDABC. European Interoperability Framework draft version 2.0, 2008. Available from http://ec.europa.eu/idabc/servlets/Docb0db.pdf?id=31597 [accessed 07 July 2011].

IDEAS. Thematic Network, IDEAS: Interoperability Development for Enterprise Application and Software-Roadmaps. Annex 1-DoW, 2002.

IDEAS. IDEAS project deliverables (WP1-WP7) (Public reports), 2003.

IEEE. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, Institute of Electrical and Electronic Engineers. International journal of service industry management, 1990, Vol. 14(2), pp. 160-172.

ISO/IEC/IEEE 42010. Systems and software engineering - Architecture description. 2011.

Jagdev H. S., Browne J. The Extended Enterprise A Context for Manufacturing. Production Planning & Control, 1998, pp. 216-229.

Johnston R., Clark G. Service operations management: improving service delivery. Financial Times, Prentice Hall, 2008, 3. Ed.

Jouault F., Allilaire F., Bézivin J., Kurtev I. ATL: A model transformation tool. Science of Computer Programming, 2008, Vol. 72, pp. 31-39.

Kleppe A. G., Warmer J., Bast W. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

Lovelock C., Wirtz J. Lapert D. Marketing des Services. Paris, France: Prentice Hall, 2004, 2nd ed., pp 619.

March J. G., Simon H. A. Les organisations. Paris, Dunod, 1958/1965

Narahari Y. Petri nets. Resonance, 1999, Volume 4, Issue 9, pp 44-52.

NIST (National Institute of Standards and Technology). IDEF Integrated DEFinition Methods. 1993. Accessed May 15 2013. http://www.idef.com/IDEF0.htm

Oliva R., Kallenberg R. Managing the transition from products to services. International Journal of Service Industry Management, 2003, Vol. 14 Iss: 2, pp.160-172.

OMG. XML Metadata Interchange (XMI). 2000, document number: ad/2001-06-12.

OMG. MDA Guide Version 1.0.1. 2003, Document number: omg/2003-05-01.

OMG. Meta-object facility 2.0 core specification. 2006, available at http://www.omg.org/spec/MOF/2.0/

OMG-1.Unified Modelling Language: infrastructure (version 2.4.1). 2011.

OMG-2. Business Process Model and Notation (BPMN) version 2.0. 2011, document number: formal/2011-01-03.

OMG. Diagram Definition (DD) Version 1.0. 2012, available at http://www.omg.org/spec/DD/1.0/ (accessed 30 Mars 2015).

Papyrus official site: http://www.eclipse.org/papyrus/.

Perrey R., Lycett M. Service-oriented architecture. In Proceedings of Symposium on Applications and the Internet Workshops, 2013.

Scheer A. W., Galler J., Kruse C. Workflow Management within the ARIS framework. In European Workshop on integrated manufacturing systems engineering Chapman & Hall, Grenoble, France, 1994.

Scheer A. W. Architecture of Integrated Information System (ARIS) In JSPE-IFIP WG 5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing (DIISM'93), 1993, Tokyo, Japan, pp. 177-191.

Scheer A. W. ARIS - Des processus de gestion au système intégré d'applications, Springer-Verlag, 2002.

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

SLACK N. Patterns of Servitization: Beyond products and service. 2005, Institute for Manufacturing, Cambridge University. London, UK.

Thoben K., Jagdev H., Eschenbaecher J. Extended Products: Evolving Traditional Product concepts. In Proceedings of the 7th International Conference on Concurrent Enterprising:

Engineering the Knowledge Economy through Co-operation. Bremen (Germany), 2001, pp. 429-439.

Thoben K. D., Jagdev H. S. Anatomy of Enterprise Collaborations Typological Issues in Enterprise Networks. Production Planning Control, 2001, pp. 437-451.

Davenport T. Process Innovation: Reengineering work through information technology. Harvard Business School Press, Boston, 1993.

Ullberg J., Chen D., Johnson P. Barriers Driven Methodology For Enterprise Interoperability. IFIP International Federation for Information Processing, 2007, pp. 453-460.

Vandermerwe S., Rada J. Servitization of Business: Adding Value by Adding Services. European Management Journal, 1988, Vol.6, pp. 314-324.

Vargo S. L., Lusch R.F. Evolving to a new dominant logic for marketing. Journal of Marketing, 2004, Vol. 68, pp. 1-17.

Vlietstra J. A summary of the CIMOSA reference architecture. In Bernus, P., Nemes, L. and Williams, T.J. (Eds.), Architectures for Enterprise Integration. London: Chapman & Hall, 1996.

Vreede G. J., Verbraeck A., Eijck D. T. Integrating the Conceptualization and Simulation of Business Processes – A Modelling Method and Arena Template. Simulation Transactions of the Society for Modelling and Simulation International, 2003, Vol. 79(1), pp. 43-55.

UNINOVA official web site: http://www.uninova.pt/

Wainer. DEVS Tools. Hosted by G. Wainer at Carlton University, November 2013, http://www.sce.carleton.ca/faculty/wainer/standard/tools.htm

Weske M. Business Process Management: Concepts, Languages, Architectures. New York, Springer-Verlag, 2007, pp. 368.

Development of GERAM, A Generic Enterprise Reference Architecture and Enterprise Integration Methodology. Integrated Manufacturing Systems Engineering, IFIP - The International Federation for Information Processing, 1995, pp 279-288.

Williams T. J. An overview of PERA and the Purdue Methodology. Architectures for Enterprise Integration, IFIP Advances in Information and Communication Technology, 1996, pp 127-161.

Zacharewicz G., Frydman C., Giambiasi N. G-DEVS/HLA Environment for Distributed Simulations of Workflows. Journal Simulation, May 2008, Vol. 84 (5), pp. 197-213.

Zaring O. Creating eco-efficient producer services. Research Report, Gothenburg Research Institute, Gothenburg, 2001.

Zeigler B. Theory of Modelling and Simulation. John Wiley & Sons, New York, 1976.

Zeigler B., Vahie S. DEVS formalism and methodology: unity of conception/diversity of application. In Proceedings of the 25th Winter Simulation Conference, pp 573–579, Los Angeles, CA, 1993.

Zeigler B., Praehofer H., Kim T. G. Theory of Modelling and Simulation - Second Edition. Academic Press, 2000.

Zelm M., Vernadat F., Kosanke K. The CIMOSA Modelling Process. Computers in Industry, 1995, vol. 27(2), pp. 123-142.

Zur Muehlen, Michael and Recker, Jan C. How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Proceedings of 20th International Conference on Advanced Information Systems Engineering, 2008, Montpellier, France

Annex-1-MetaModels

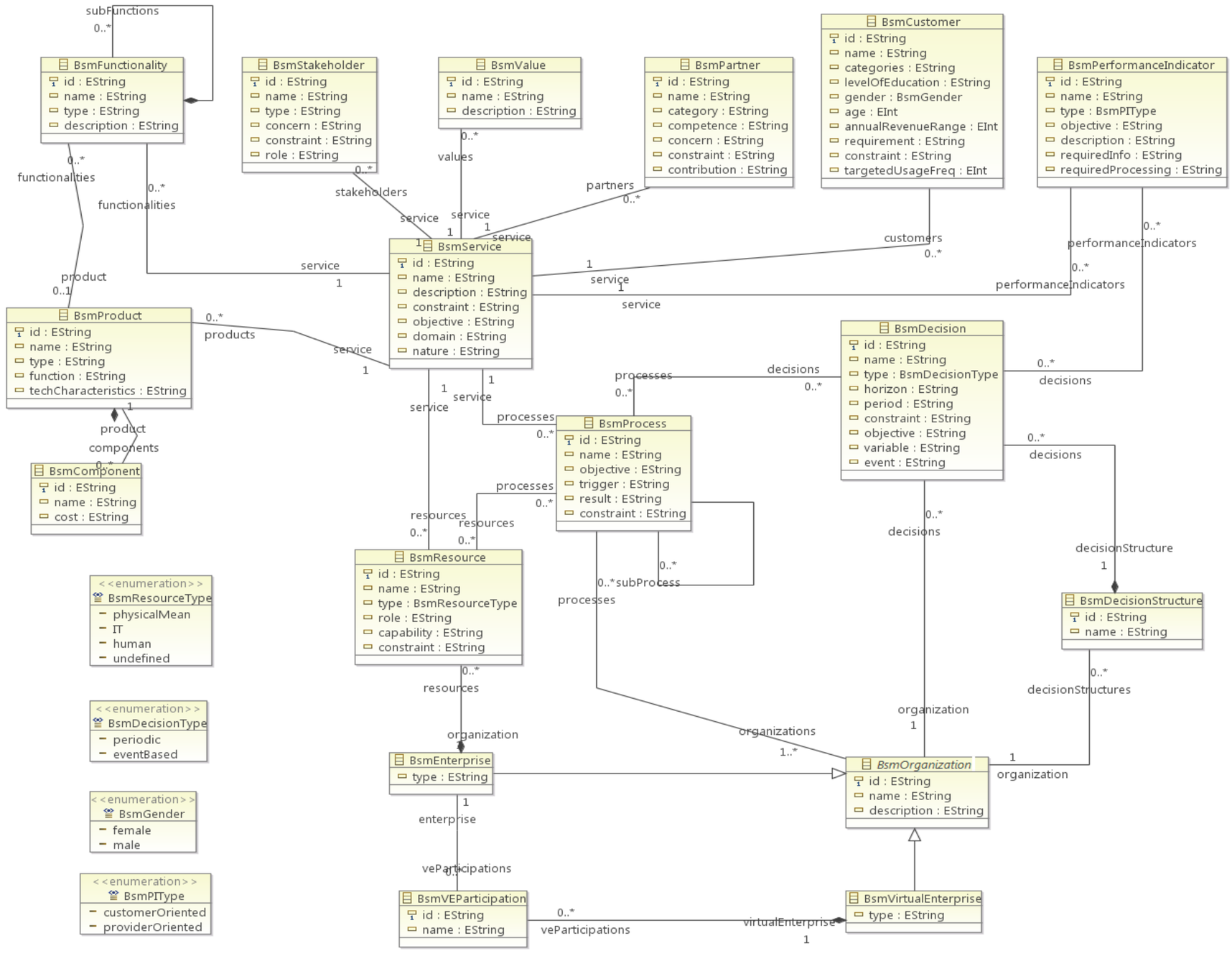


Figure 83 BSM Core Metamodel

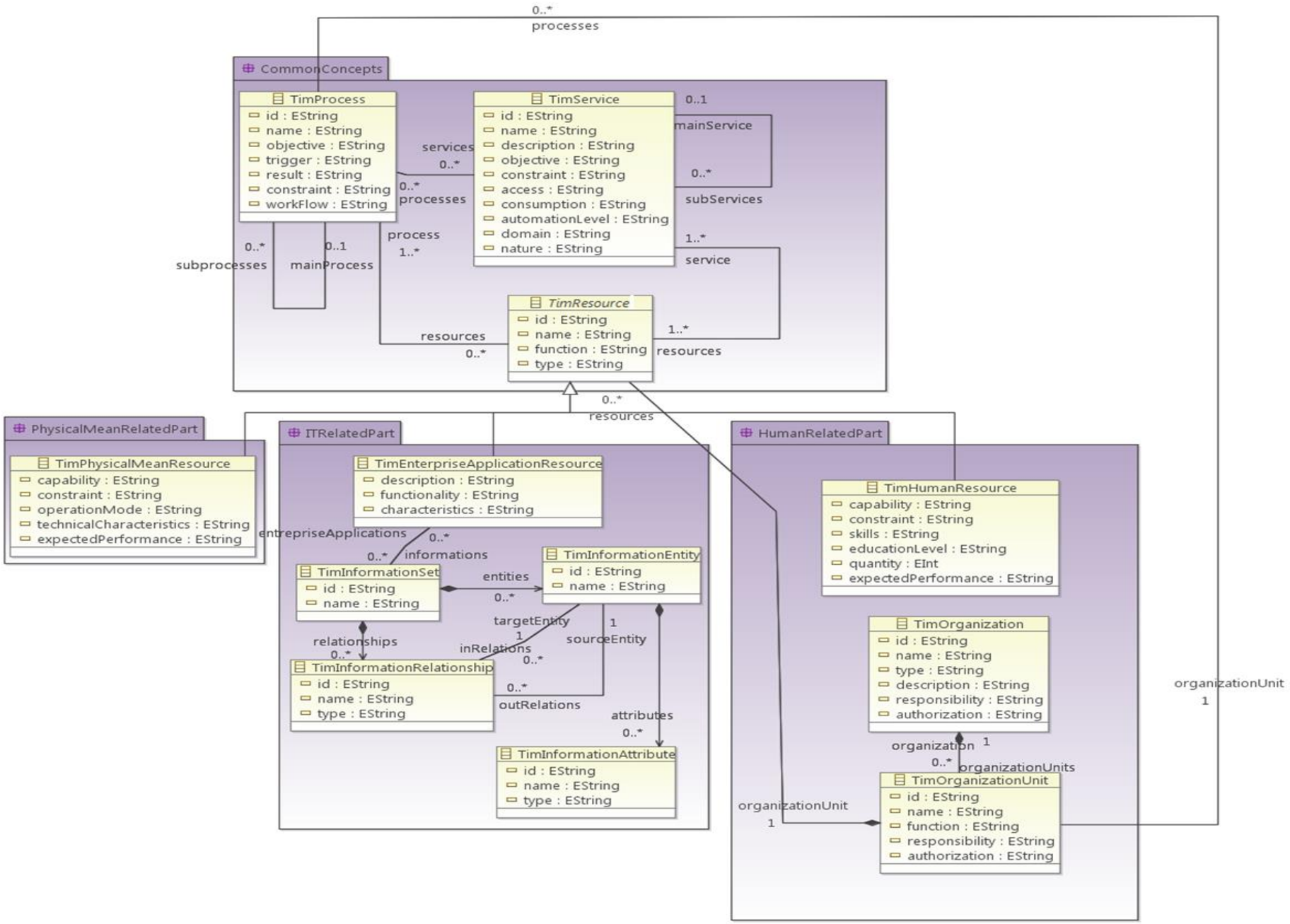


Figure 84 TIM Core Metamodel

Annex-2-Simulation Report

Simulation Report

produced by the SLMToolBox

Simulation Profile

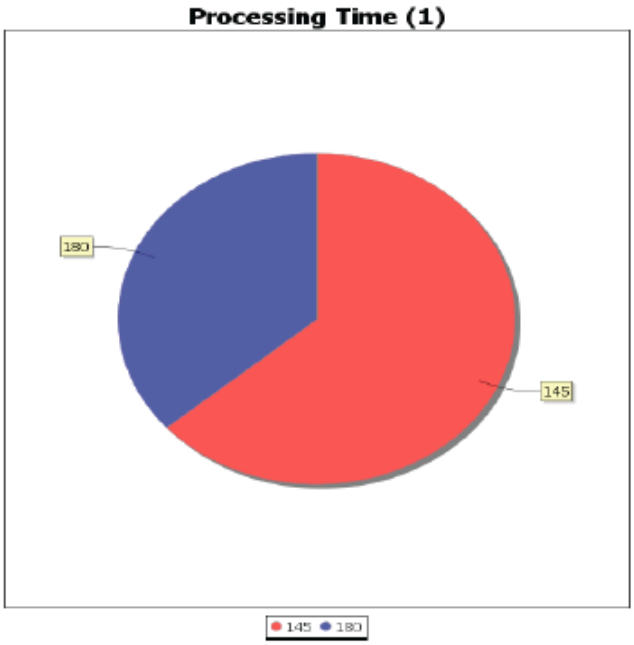
Number of instances: 30

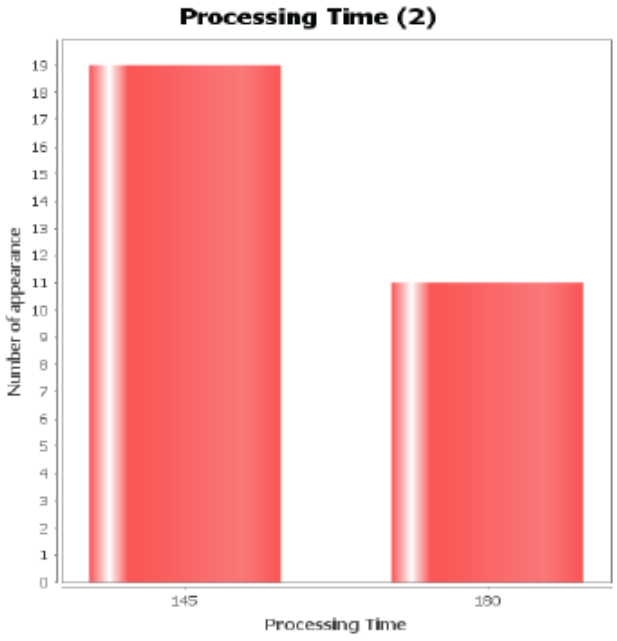
User's Objective:

- Optimal Processing Time waited by the user: 200
- Optimal Cost waited by the user: 160

Processing Time

The "Processing Time" Pie graph and bar graph below view the distibution of different Processing time obtained throught the simulation process.





Highest Processing Time

The highest processing time obtained during the simulation is 180

Path 0: capture incident ---> classify incident ---> technical incident ---> send message

Lowest Processing Time

The lowest processing time obtained during the simulation is 145

Path 0: capture incident ---> classify incident ---> functional incident ---> send message

Most Probable Processing Time

The most probable processing time obtained during the simulation is 145

Path 0: capture incident ---> classify incident ---> functional incident ---> send message

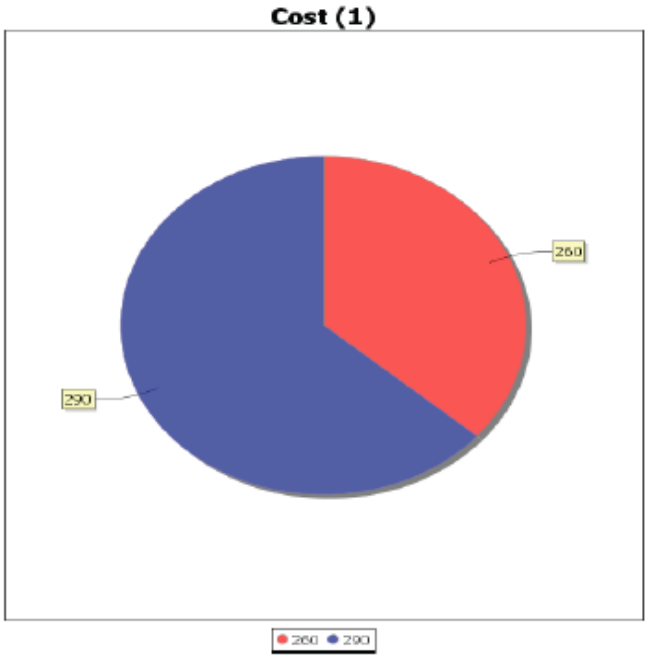
Least Probable Processing Time

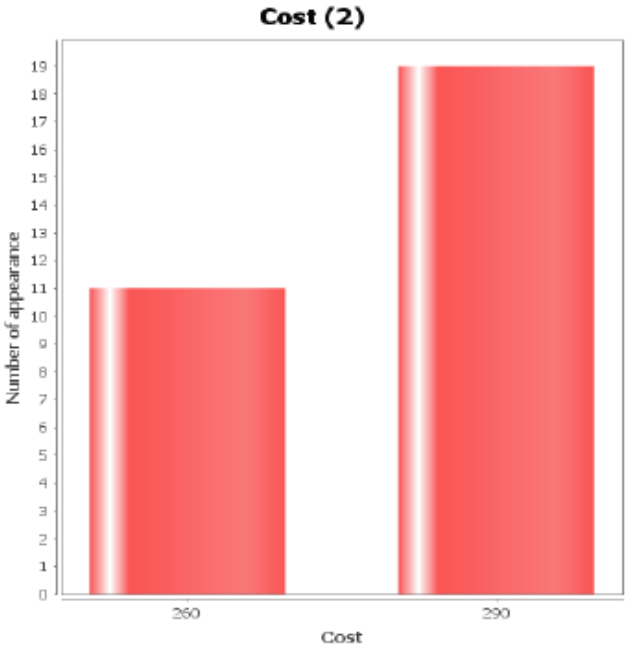
The lest probable processing time obtained during the simulation is 180

Path 0: capture incident ---> classify incident ---> technical incident ---> send message

Execution Cost

The "Cost" Pie graph and bar graph below view the distibution of different Costs obtained throught the simulation process.





Highest Execution Cost

The highest execution cost obtained during the simulation is 290

Path 0: capture incident ---> classify incident ---> functional incident ---> send message

Lowest Execution Cost

The lowest execution cost obtained during the simulation is 260

Path 0: capture incident ---> classify incident ---> technical incident ---> send message

Most Probable Execution Cost

The most probable execution cost obtained during the simulation is 290

Path 0: capture incident ---> classify incident ---> functional incident ---> send message

Least Probable Execution Cost

The least probable execution cost obtained during the simulation is 260

Path 0: capture incident ---> classify incident ---> technical incident ---> send message

Annex-3-ATL and XSLT code

```
-- ------------------------------------------
-- transform an EA* Process element to a BPMN Process element
lazy rule ProcessToProcess {
    from
        s: EA!EaProcess (
            s.oclIsTypeOf(EA!EaProcess)
        )
    to
        a: BPMN!Process (
            id <- s.id,
            name <- s.name,
            flowElements <- s.flowElements.
            append(thisModule.createUserTasks(thisModule.getActivities_HumanRes(s.flowElements -> select (e|e.oclIsTypeOf(EA!EaSupportFlow))))).
            append(thisModule.createServiceTasks(thisModule.getActivities_ITRes(s.flowElements -> select (e|e.oclIsTypeOf(EA!EaSupportFlow))))).
            append(thisModule.createTasks(s.flowElements -> select(e | e.oclIsKindOf(EA!EaExtendedActivity)) -> select(e | (thisModule.getActivities_HumanRes(s.flowElements -> select
            (e|e.oclIsTypeOf(EA!EaSupportFlow))).union(thisModule.getActivities_ITRes(s.flowElements -> select (e|e.oclIsTypeOf(EA!EaSupportFlow))))).excludes(e)))
            )
            .append(s.flowElements -> select (e| e.oclIsTypeOf(EA!EaOr))-> collect (e| thisModule.Or2ExclusiveGateway(e)))
            .append(s.flowElements -> select (e| e.oclIsTypeOf(EA!EaAnd))-> collect (e| thisModule.And2ParallelGateway(e)))
            .append(s.flowElements -> select (e| e.oclIsTypeOf(EA!EaMaterialResource)) -> collect (e| thisModule.Material2DataObject(e)))
            .append(s.flowElements -> select (e| e.oclIsTypeOf(EA!EaProcessConnector)) -> collect (e| thisModule.ProcessConnector2CallActivity(e)))
            .append(thisModule.createSubProcess(s.flowElements -> select(e | e.oclIsTypeOf(EA!EaStructuralExtendedActivity)))),
            laneSets <- thisModule.laneSet
        )
    do{
        let s: String = '' in a.name.toString().println();
        thisModule.bpmnProcess <- a;
        thisModule.bpmnProcess.flowElements <- thisModule.bpmnProcess.flowElements.union(thisModule.bpmnFlowElements);
        thisModule.bpmnProcessRef <-a;
        thisModule.eaStarProcessRef <- s;
        thisModule.laneSet.lanes <- thisModule.laneSet.lanes.union(thisModule.createLanes(thisModule.getResponsibleResources(s.flowElements -> select (e| e.oclIsTypeOf(EA!EaSupportFlow))))).
        union(thisModule.createLanesForOrganization(s.model.organizations.excluding(s.model.organizations.first())));
    }
}
```

Figure 85 ATL Lazy Rule: EA* Process to BPMN Process

```
-- get all Extended Activities with IT resources
helper def: getActivities_ITRes(eaFlows : Set(EA!EaSupportFlow)): Set(EA!EaExtendedActivity) =
    eaFlows -> select (e | e.source.oclIsTypeOf(EA!EaITResource) and e.resourceRole.toString().startsWith('res')) -> collect (e | e.target).asSet();

-- create UserTasks and test if it is a starting or ending task
helper def: createUserTasks(activities: Set(EA!EaExtendedActivity)): Set(BPMN!UserTask) =
    activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2UserTask(e)).union(
        activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and  e.isEnding and  e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingEndingUserTask(e))).
        union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingUserTask(e))).
        union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2EndingUserTask(e)));

-- create ServiceTasks and test if it is a starting or ending task
helper def: createServiceTasks(activities: Set(EA!EaExtendedActivity)): Set(BPMN!ServiceTask) =
    activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2ServiceTask(e)).union(
        activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and  e.isEnding and  e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingEndingServiceTask(e))).
        union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingServiceTask(e))).
        union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2EndingServiceTask(e)));

-- create tasks
helper def: createTasks(activities: Set(EA!EaExtendedActivity)): Set(BPMN!Task) =
    activities -> select(e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2Task(e)).
    union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and  e.isEnding and  e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingEndingTask(e))).
    union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and not e.isEnding and e.isStarting) -> collect(e | thisModule.ExtendedActivity2StartingTask(e))).
    union(activities -> select (e | e.oclIsTypeOf(EA!EaAtomicExtendedActivity) and e.isEnding and not e.isStarting) -> collect(e | thisModule.ExtendedActivity2EndingTask(e)));

-- get all resources which are responsible for the execution of activities
helper def: getResponsibleResources(flows: OrderedSet(EA!EaSupportFlow)): OrderedSet(EA!EaResource) =
    flows -> select(e | e.resourceRole.toString().startsWith('res') and thisModule.isFirstOrganization(e.target.organization)) -> collect ( e| e.source).asOrderedSet();

helper def: isFirstOrganization(organization: EA!EaOrganization): Boolean =
    if (thisModule.eaStarProcessRef.model.organizations.indexOf(organization) <= 1 )
        then true
    else false
    endif;
-- create lanes with respect to responsible resources
helper def: createLanes(resources: Set(EA!EaResource)): Set(BPMN!Lane) =
    resources -> collect(e| thisModule.Resource2Lane(e));
```

Figure 86 ATL helpers

```
<!-- create pools  -->
<xsl:for-each select="rootElements">
    <xsl:if
        test="substring-after(@xmi:type, ':') = 'Collaboration' or substring-after(@xsi:type, ':') = 'Collaboration'">
        <xsl:for-each select="participants">
            <xsl:variable name="idForMessage">
                <xsl:value-of select="@id" />
            </xsl:variable>
            <bpmndi:BPMNShape>
                <xsl:attribute name="isHorizontal">
                    <xsl:text>true</xsl:text>
                </xsl:attribute>
                <xsl:attribute name="id">
                    <xsl:text>BPMNShape_Participant_</xsl:text>
                    <xsl:number expr="position()" />
                </xsl:attribute>
                <xsl:attribute name="bpmnElement">
                    <xsl:value-of select="@id" />
                </xsl:attribute>
                <dc:Bounds>
                    <xsl:attribute name="height">
                    <xsl:if test="substring-before(@processRef, '_') = 'EaProcessImpl'">
                    <xsl:value-of select="$numberLanes * 200" />
                    </xsl:if>
                    <xsl:if test="substring-before(@processRef, '_') != 'EaProcessImpl'">
                        <xsl:value-of select="200" />
                    </xsl:if>
                    </xsl:attribute>
                    <xsl:attribute name="width">
                        <xsl:text>1030.0</xsl:text>
                    </xsl:attribute>
                    <xsl:attribute name="x">
                        <xsl:text>170</xsl:text>
                    </xsl:attribute>
                    <!-- the y of the pool is decided with respect to its position and the type of the process referenced -->
                    <xsl:attribute name="y">
                        <xsl:if test="substring-before(@processRef, '_') = 'EaProcessImpl'">
                            <xsl:value-of select="100" />
                        </xsl:if>
                        <xsl:if
                        test="substring-before(@processRef, '_') != 'EaProcessImpl'">
                            <xsl:variable name="num">
                                <xsl:number expr="position()" />
                            </xsl:variable>
                            <xsl:value-of
                        select="((number($num) -1 )* 200)+ (number($num) * 100.0) + ($numberLanes * 200) + 100 " />
                        </xsl:if>
                    </xsl:attribute>
```

Figure 87 XSLT example 1

```
    <xsl:for-each select="relationships">
        <bpmn2:relationship>
            <xsl:call-template name="RelationshipAttributesTemplate" />
            <xsl:call-template name="RelationshipTemplate" />
        </bpmn2:relationship>
    </xsl:for-each>
```

Figure 88 XSLT example 2

Annex-4 Use case diagrams

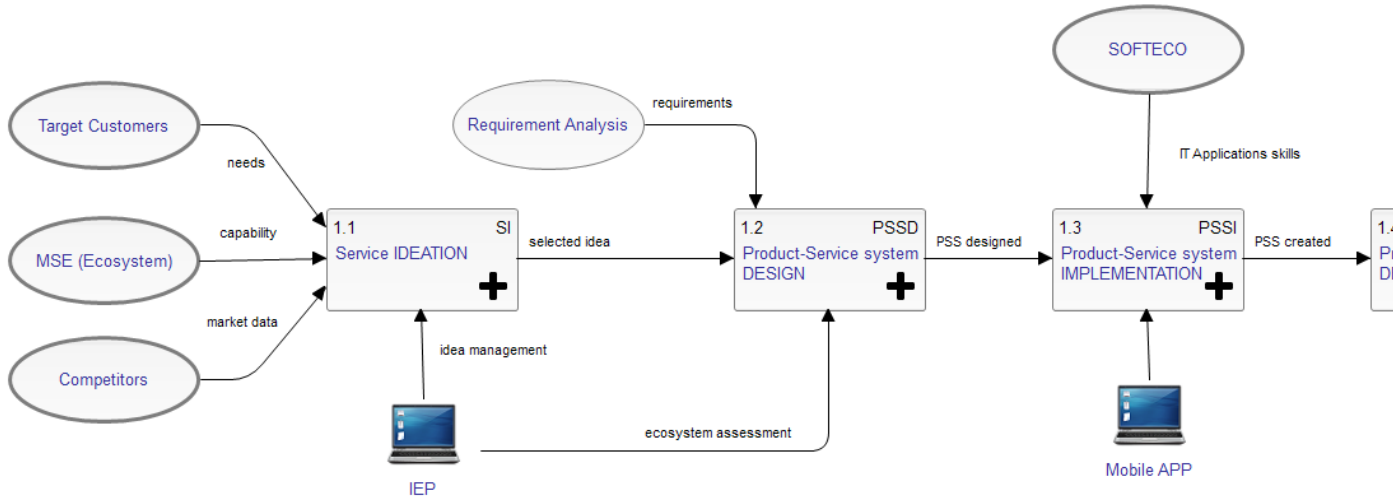


Figure 89 Global view

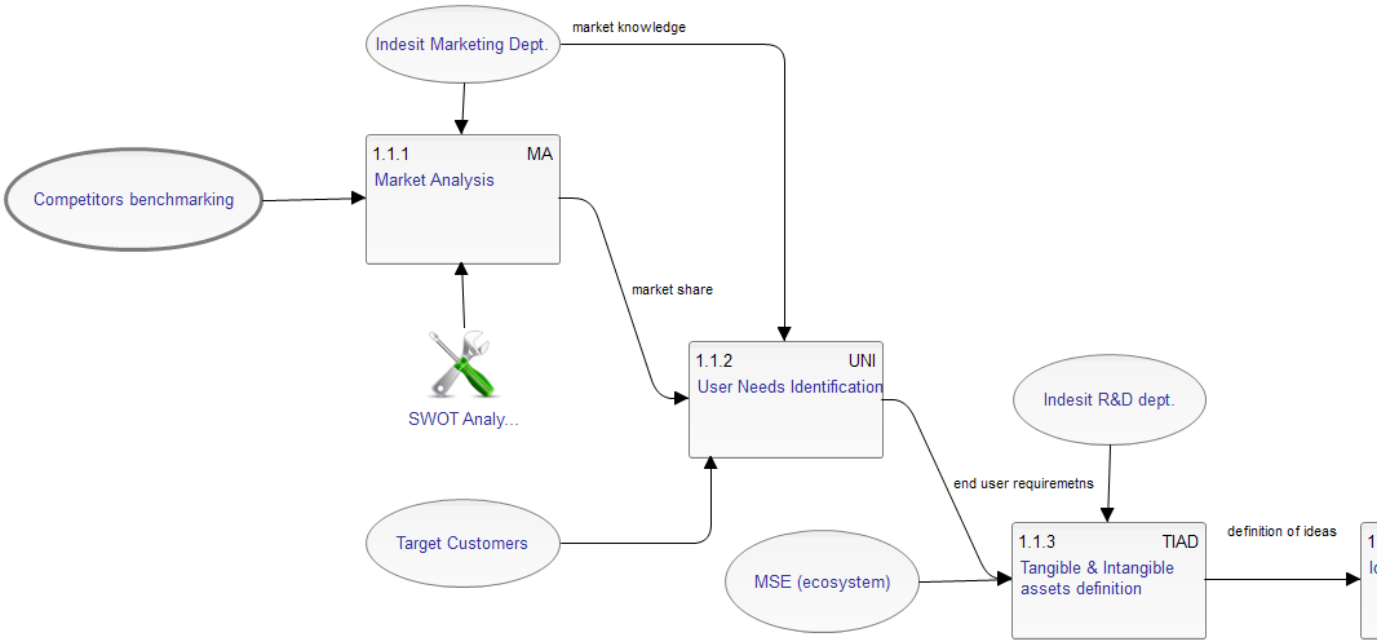


Figure 90 Service Ideation process

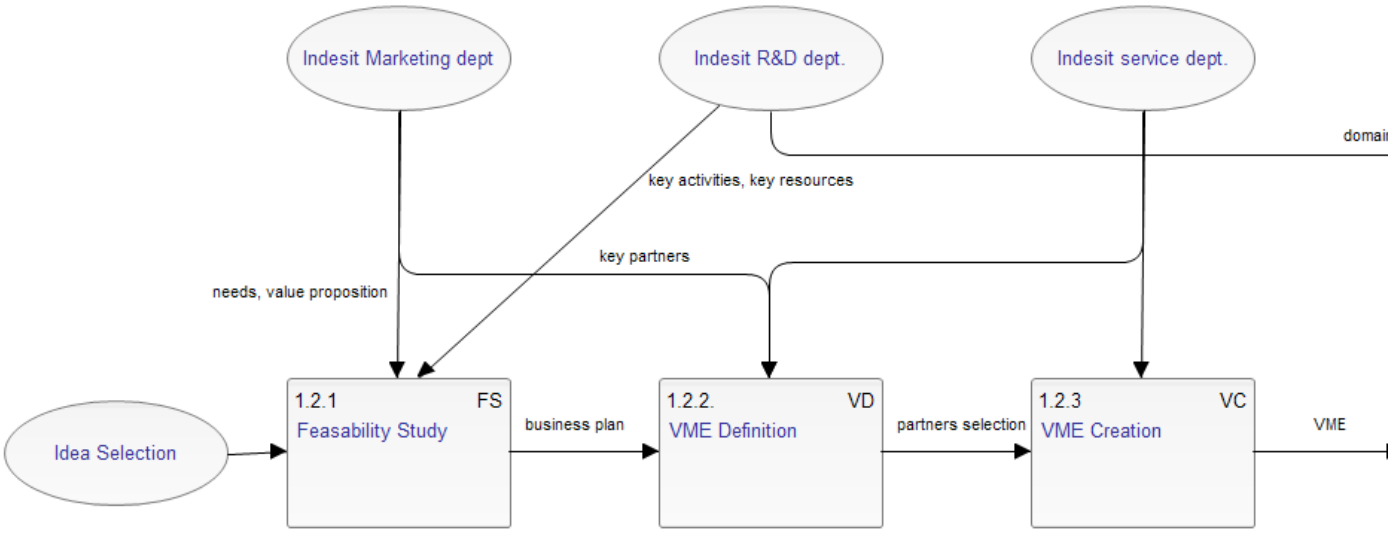


Figure 91 Product-Service System Design

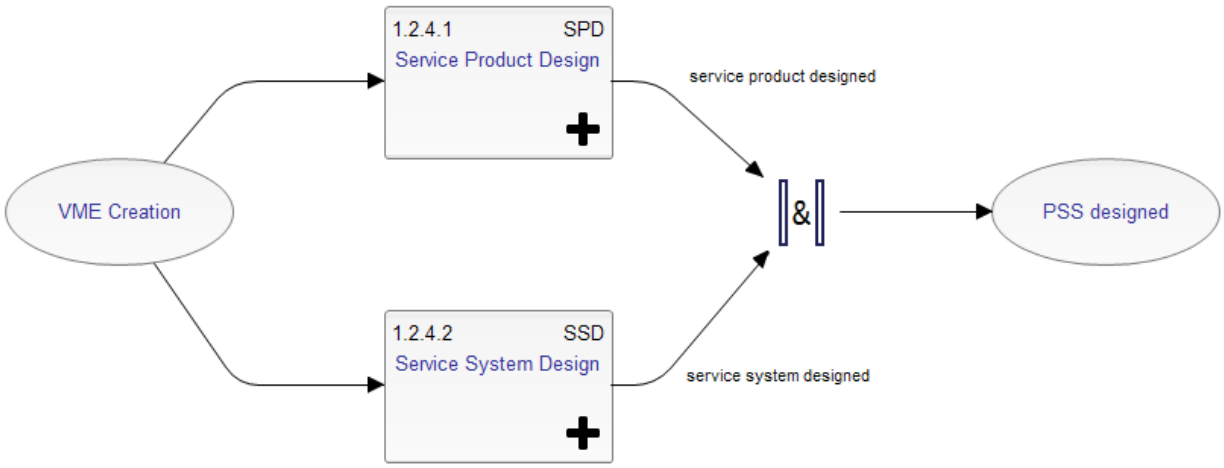


Figure 92 Product and Service Design

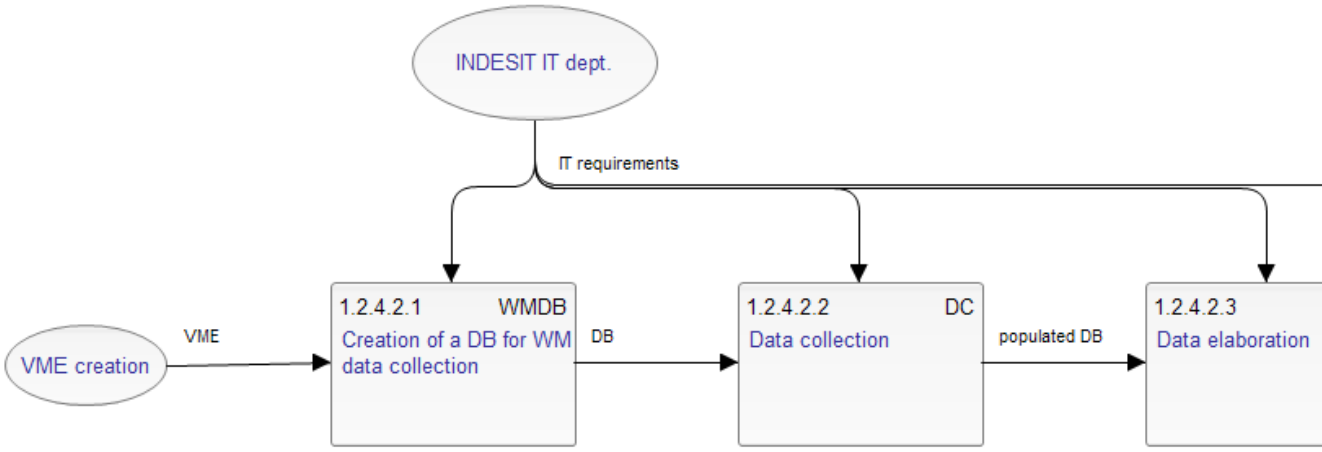


Figure 93 Service System Design

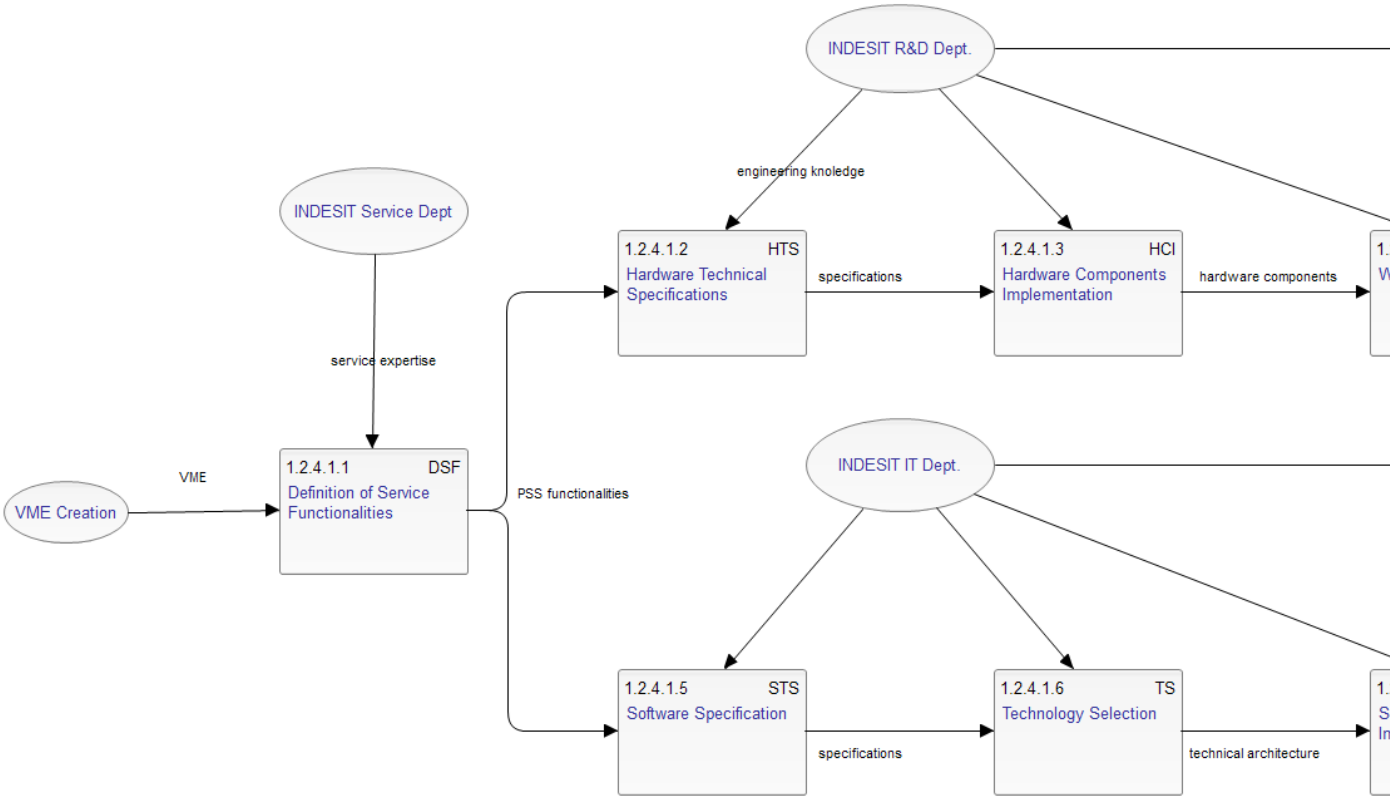


Figure 94 Service System Design