Information Models and Transformation Principles Applied to Servitization of Manufacturing and Service Systems Design

Carlos Agostinho[1], Hassan Bazoun[2, 3], Gregory Zacharewicz[2], Yves Ducq[2] and Hadrien Boye[3]
[1]*Centre of Technology and Systems, CTS, Uninova, 2829-516, Caparica, Portugal*
[2]*University of Bordeaux – IMS/LAPS, 33405, Talence Cedex, France*
[3]*Hardis Ouest, 44800, Saint Herblain, France*

Keywords: MDA, MDE, Enterprise Interoperability, Model Transformations, MDSEA, Manufacturing Servitization.

Abstract: Based on systems engineering principles, information modelling is seen as a central activity for the development and life cycle support of a product or system. It enables to reduce the costs derived from miscommunications and misconceptions normally occurring throughout the service system design, analysis and maintenance activities. Supporting the servitization of manufacturing and the evolution towards product-service systems or extended products, modelling and interoperability is becoming of utmost importance to ensure coherence among conceptual design phases at organizational levels down to technology development. Therefore, this paper explores the model-driven development and model-driven interoperability transformations principles to unify every step of the development of service systems, from its start at the application's business requirements, through the design of technology independent functions, to deployable services.

1 INTRODUCTION

The evolution from an economy of products towards an economy of services has been becoming important in manufacturing since the nineties. The "Servitization" concept, intrinsically linked to discussions on services and service provision, is loosely defined around the delivery of product-based services, and its most tangible effect is the development of Product Service Systems (PSS) and Extended Product (Vandermerwe & Rada, 1988; Baines et al., 2009; Thoben et al., 2001).

Being the application of competence for the benefit of another, a service involves at least two entities to enable value co-creation (Spohrer et al., 2007). This service orientation trend has been gaining momentum also in ICT to support the integration of products and services with customers. They are not just provided with products, but broader more tailored solutions based on the customer centricity paradigm (Baines et al., 2009). In fact, service-oriented architectures have modernized information systems towards that direction, providing powerful methods and tools to decompose complex systems in autonomous components, and supporting enterprise processes and workflows with simple orchestrations and compositions (Ducq, Doumeingts, et al., 2012).

Embracing the servitization integrated view on PSS and extended products, services will concern physical products as well as the associated technology, people and knowledge (Chesbrough & Spohrer, 2006). Indeed, 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 a common goal, sharing risks and resources (Ducq, Chen, et al., 2012). This requires the integration of autonomous, geographically distributed and heterogeneous stakeholders in virtual organizations and business ecosystems, creating, sharing and reusing information across teams and enterprise boundaries (Zdravkovic et al., 2013).

Therefore, to meet the above requirements, there is a need to apply modelling paradigms in order to support Service Systems (SS) development along the lifecycle of different organizational forms such as Virtual Manufacturing Enterprise (VME) (Ducq, Doumeingts, et al., 2012; Estefan, 2007). Nevertheless, developers need to take care of properties such as interoperability. Being directly related with the heterogeneity of modelling

Agostinho C., Bazoun H., Zacharewicz G., Ducq Y. and Boye H..
Information Models and Transformation Principles Applied to Servitization of Manufacturing and Service Systems Design.
DOI: 10.5220/0004875206570665
In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development* (MDSE-2014), pages 657-665
ISBN: 978-989-758-007-9

languages, communication capabilities, databases and semantics, interoperability hides a great barrier in the path towards collaborative services development. In fact, since many organizations within VME's and enterprise networks use software solutions based on their own needs, the cooperation with others is not a trivial activity (Jardim-Goncalves et al., 2013).

To solve this problem, the authors propose Model Driven Architecture (MDA) based technologies such as Model-Driven Development (MDD) and Model-Driven Interoperability (MDI) to unify every step of the development of service systems, from its start specifying application's business requirements, through the design of technology independent functions and behaviour, to deployable services. Based of these principles, and continuing the work of Ducq et al. (2012), this paper explores a methodology for service system design and implementation, and proposes a framework for the specification of mappings and execution of automatic transformations among different models and modelling levels. It enables to respond to the service lifecycle and VME dynamics, ensuring its sustainability along service (re)engineering and co-design, i.e. changes that occur over time and could impact negatively the business ecosystem can be controlled, tuned and balanced to maximize servitization efficiency without jeopardizing interoperability.

2 MODEL DRIVEN SERVICE SYSTEM ENGINEERING

Service systems emphasize collaboration and adaptation in value co-creation, and establish a balanced and interdependent framework for systems of reciprocal service provision. Such systems may be business entities that survive, adapt, and evolve through mutual exchange and application of resources – particularly knowledge and skills (Spohrer et al., 2007). SS engage in exchange with others to enhance adaptability and survivability, co-creating value for both. All these are issues related to the Enterprise Interoperability (EI) domain, thus some EI intensive concepts and methods, such as modelling, can be adapted to service systems engineering (Agostinho, Jardim-Goncalves, et al., 2012; Jardim-Goncalves et al., 2012).

Also, being a hot topic for the last couple of years, service management derived from product lifecycle management, aiming at handling all service data relating to its design, implementation, operation and final disposal (Garschhammer et al., 2001). Based on ISO 15704 (ISO TC184/SC5, 2000), the various service system engineering phases iterate among: (1) identification, (2) concept, (3) requirement, (4) design, (5) implementation, (6) operation and (7) decommission. A service could be re-engineered several times during its life, and feedback loops could happen in order to answer better to the requirements of the previous phase (Ducq, Doumeingts, et al., 2012).

In this context, service modelling seeks to formalise the concept of a service, largely through definition on the participants in service value creation (providers and consumers). Proposed models include those by Garschhammer et al. (2001), and Kohlborn et al. (2009) generic business service management framework, form the early engineering phases, and follow model-driven principles to iterate through the different phases.

2.1 Model Driven Engineering (MDE) and Architecture (MDA)

MDE, sometimes also referred as model-driven development, is an emerging practice for developing model-driven applications. Popularized by the OMG MDA (OMG, 2003), it represents a promising software engineering approach to address systems complexity, by simplifying and formalizing the various activities and tasks that comprise an information system life cycle. MDE is meant to maximize compatibility between systems, simplifying the process of design, and promoting communication between teams working on the system (Selic, 2003; Agostinho, Černý, et al., 2012).

MDD/MDE's vision encourages the use of models at different levels of abstraction, from high-level business models focusing on goals, roles and responsibilities down to detailed use-case and scenario models for business execution (Bézivin, 2005; Frankel, 2003). These models are developed through extensive communication among product managers, designers, and members of the development team, and as they approach completion, enable a fast development of systems.

An MDA system can be observed and analysed from different points of view, defining a hierarchy of models at three different levels of information abstraction (OMG, 2003): (a) Computation Independent Model (CIM), specifying the requirements and the environment where the system will operate. It is meant for the domain practitioners and is based on the vocabulary of the specific target domain; (b) Platform Independent Model (PIM),

formalising the structure and functionality of the system. This model focuses on operational details while hiding specific details of any particular platform in order to be suitable for use with several different platforms; and (c) Platform Specific Model (PSM) that combines the PIM model with implementation constructs that specify how the system uses a particular type of platform.

Model transformation methods are used to automate the translation process from high level specifications (CIM) and formal descriptions of the systems (PIM), to the bottom levels (PSM) and implementation code, increasing speed, code optimization and avoiding errors in the engineering process (Frankel, 2003; Agostinho, Černý, et al., 2012).

2.2 Model-Driven Service Engineering Architecture (MDSEA)

Based on the above MDA principles, the MDSEA architecture is being defined along the MSEE integrated project (www.msee-ip.eu/) to model and guide the transformation from the business requirements of the SS into detailed specifications of components that must be implemented to support the servitization process (Ducq, Chen, et al., 2012; Ducq, Doumeingts, et al., 2012).

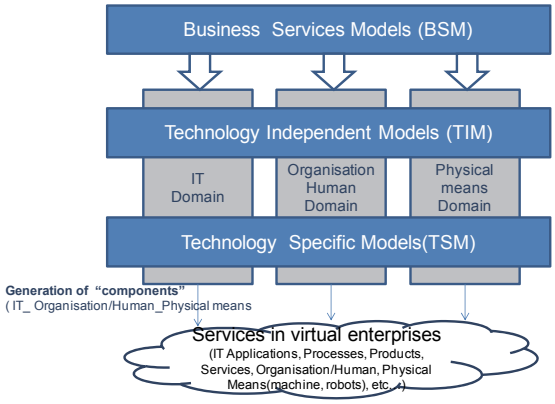

Figure 1: MDSEA (Ducq, Chen, et al., 2012).

As illustrated in Figure 1 and in resemblance to the MDA's CIM-PIM-PSM, this architecture defines several modelling levels to have a progressive specification of service system components at the business (Business Service Modelling - BSM), functional (Technology Independent Modelling - TIM), and technological (Technology Specific Modelling - TSM) levels. Extending the modelling approach of MDA with principles explored on EI's enterprise modelling, technology related models integrate not only the IT part but also the requirements leading to the implementation of a solution in organization and physical domains.

The approach implies that the different model, obtained via model transformation from the upper-level ones, should use dedicated service modelling languages that represent the system with the appropriate level of description. GRAI Integrated Modelling (Chen & Doumeingts, 1996) and BPMN 2.0 (OMG, 2011a) have been considered as a reference for the BSM and TIM levels, but further details on the analysis and selection of the most appropriate languages can be found in MSEE (2012); Ducq et al. (2012).

3 METHODOLOGY FOR SERVICE SYSTEM DESIGN & IMPLEMENTATION

In order to operationalize enterprises servitization using model-based engineering and interoperability concerns presented before, it is necessary to propose a precise method to implement the service system. Figure 2 illustrates a structured multi-step approach to-be implemented through various groups of actors belonging to the enterprises and organizations involved in the servitization process, as well as by external actors to support this process.

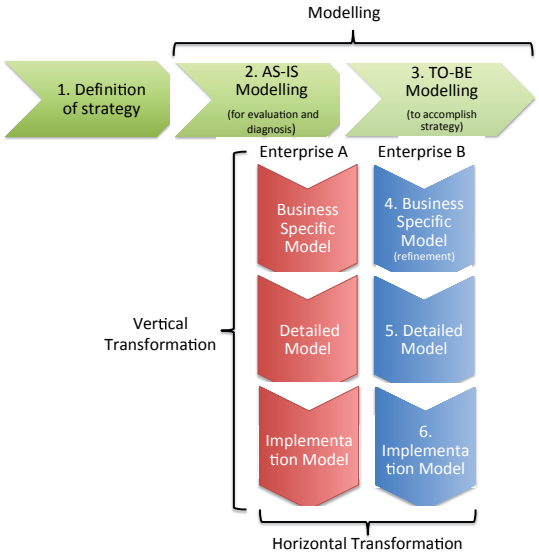

Figure 2: MSEE Methodology for servitization system definition and implementation.

The methodology begins at the strategic level of companies that want to evolve towards service-

oriented business methods following the extended product stages introduced by Thoben et al. (2001), i.e. from a single product, to product with services, until product-as-a-service. Depending on their objectives, modelling and MDSEA vertical transformations are required to go from the desired strategy, a "to-be" business specific model (BSM level), towards a detailed functional definition (TSM) and practical implementation model (TSM). Moreover, since models can be shared to enable value co-creation (e.g. through collaborative process orchestration) among companies operating at several domains and using different technologies, horizontal transformations are also considered to ensure interoperability. In summary, the horizontal flow initializes the study to reach the "to-be", while the vertical sequence allows to implement MDSEA and determine the components of the SS by domains.

After the specification of the methodology, which contributed to the identification of concrete transformation requirements, the MDSEA architecture provided the building blocks for VME service development, scoping the work to be implemented:

- The capability to transform a business specific model into a functional one that can then be complemented by a system architect;
- The capability to transform a functional model into a technology specific one envisaging the generation of concrete software and services;
- The capability to harmonize models specified by different enterprises, enabling interoperability and collaboration (e.g. process orchestration, service matching) within the ecosystem.

4 MDSEA MODEL TRANSFORMATIONS FRAMEWORK

The MSEE methodology applies the distinction between vertical and horizontal transformations, providing interoperability and portability at the same degree of relevance as the traceability features, linking requirements, design, analysis, and testing models of the several MDSEA levels. In this context, a framework for MDSEA transformations along 3 different axes is here proposed (Figure 3):

- **Axis 1 - Modelling levels**. Defined according to the meta-modelling reference architecture proposed by OMG (OMG, 2011b), which envisages that real world data is modelled using four levels that go for data itself (M0) to the meta-meta-model (M3). Following this axis, service models are described at the level M1 using the modelling language concepts and constructs defined at level M2.
- **Axis 2 - MDSEA levels**. MDSEA enables service system modelling around the three abstraction levels summarized before in section 2.2.
- **Axis 3- VME integration**. Starting from a minimum of two systems, this axis represents the integration among the multitude of systems part of the enterprise service ecosystem.

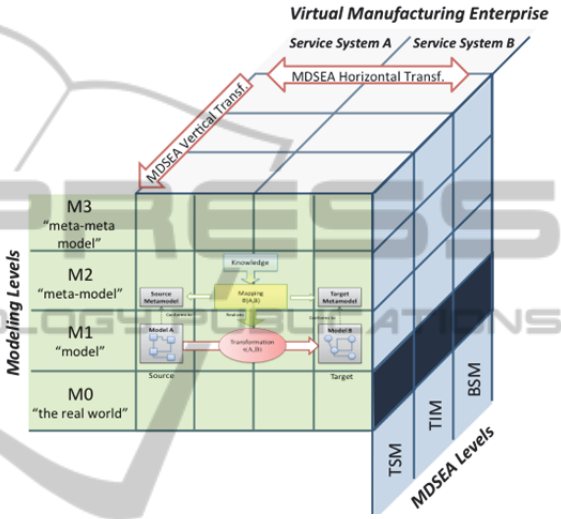

Figure 3: MDSEA Transformations Framework.

The transformations framework envisages a formal specification of models along the first axis to enable vertical transformations from BSM to TSM (axis 2) as well as horizontal ones to integrate different service systems (axis 3). Indeed, based on model transformations, the MDSEA transformations framework unifies every step of the service system development.

4.1 Vertical and Horizontal Transformations

Vertical transformations imply a change on the abstraction level of the resulting model, i.e. going from TIM to TSM implies a specialization transformation along the MDSEA axis. As in MDA vertical transformations, the amount of generated models depends on both the code generator and also the level of detail represented in the upper levels. Ideally, only small portions of missing knowledge should have to be added by the human in order to ensure that, at the TSM level, the generated code and auxiliary files are ready for compilation, linking and

deployment.

In the case of horizontal transformations, the level of abstraction remains unchanged, leading to solutions for that enable collaborative activities at the VME axis. Greater interoperability benefits are expected with this type of transformations, but as studied by Agostinho et al. (2012), one might need to be concerned also with language-related specificities. In fact, different languages enable to describe the same objects with different details (e.g. properties, constraints). Thus, both input and output models must be an instance of a well defined meta-model according to axis 1.

4.2 Transformations Architecture

Recalling the first axis of the framework, namely the relationship between the concepts of model and meta-model, when performing a model transformation (from ModelA to ModelB), one is converting instances of a source model to instances of another model, the target, and an explicit or an implicit mapping at the same meta-modelling level has to be performed.

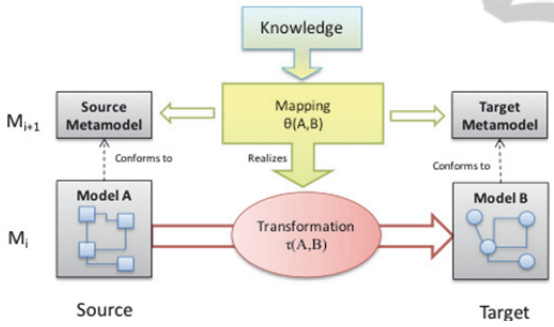

Figure 4: Generic Transformations Architecture - adapted from MDA Guide (OMG, 2003).

As depicted in the generic transformations architecture, included in the framework (frontal pane of Figure 3) and highlighted in Figure 4, the idea is that when performing a transformation "τ(A,B)" at a certain level "i", this transformation has (implicitly or explicitly) to be designed by taking into account mappings "θ(A,B)" at level "i+1". Once the "i+1" level mapping is complete, executable languages such as the Atlas Transformation Language (ATL, www.eclipse.org/m2m/atl/), the Query View Transformation (QVT, www.omg.org/spec/QVT/), or the AToM (http://atom3.cs.mcgill.ca) can be used to implement the transformation itself, either vertically along axis 2 or horizontally along axis 3.

4.3 Knowledge Management in the Transformation Process

Simple type mappings are generally insufficient to specify a complete transformation (Truyen, 2006). Additional knowledge is frequently required to complement the mapping, specifying that certain concepts in the source model must be annotated (marked) in a specific way in order to produce the desired output in the target model. Sometimes, this extra information cannot be determined from the source model itself, and it might need to use knowledge from external models, e.g. ontologies. For these reasons, the generic transformations architecture adopted by MDSEA is complemented with a "knowledge" box on top of the meta-model mappings.

Semantics are recognized as an important area for models alignment identified as one of the levels of interoperability to consider within an enterprise (Athena IP, 2006). However, a general observation shows that in traditional MDA/MDI transformations, semantic knowledge is often not exploited to improve interoperation automation.

MSDEA transformations framework envisages to change that explicitly associating that semantic knowledge (e.g. annotations, mismatches, reconciliation rules, etc.) to models and mappings. It needs syntactic alignment as a pre-requisite, so that the approach for processing the information will be interpretable from a known structure. However, once the syntactical correctness has been verified, semantic interpretation, which goes beyond syntax or structure, must be understood and unambiguously defined based on the context of the mapping definition.

5 APPLICATION IN AN INDUSTRIAL DOMAIN

Figure 5 illustrates a scenario from the clothing industry (in the frame of MSEE project) indicating the steps taken as part of the service system implementation, namely the vertical transformation of a modelled online product configurator (BSM level) into a collaboration model (TIM level) that enables specific VME services at TSM level.

More specifically, the scenario envisages to start representing of the BSM processes using the Extended Actigram Star (EA*) language (Doumeingts et al., 2006). Due to transformation requirements that model should be manually

decomposed (steps 1 and 2) into two separate models: user centred and system centred. Subsequently, automatic vertical transformation is applied to both (steps 3 and 4), and the result consists in two separate BPMN diagrams, which are also user and system centred. Then, the TIM models are merged for orchestration in order to represent the user-system interaction in one collaboration model (step 5). For simplification reasons, no horizontal transformation is required here. Later this diagram is enriched manually at the TIM level for a more detailed model and the vertical transformation continues down to the TSM level (step 6).

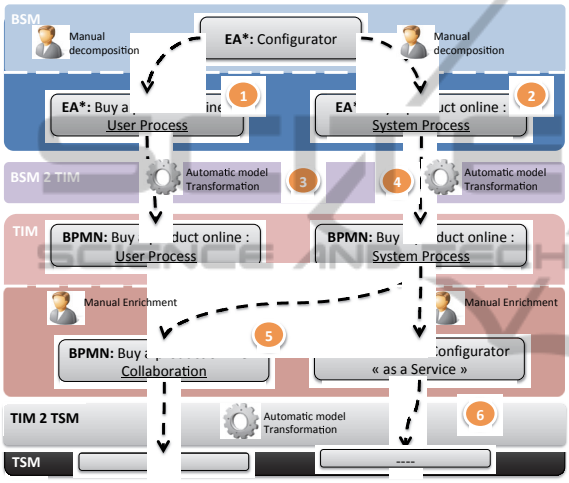

Figure 5: Modelling and transformation scenario.

Focusing on the paper contribution and due to space constraints, detailed technical examples are only taken at the transformation steps 3 and 4, helping to illustrate the mechanism. To note that this example is complying with a first priority specifications and no work with ontologies and knowledge annotation has yet been implemented for the knowledge management.

5.1 MDSEA Axis Implementations: EA* to BPMN2.0

Vertical transformations implemented along the MDSEA axis of the transformations framework require the definition of model mappings among MDSEA core concept meta-models (Ducq, Doumeingts, et al., 2012), as well as among selected languages at BSM, TIM, and TSM. Being a process modelling language, EA* language shares common direct or indirect concepts with BPMN2.0. As a result, a mapping is established from EA* concepts to BPMN2.0 concepts following specific conditions.

In the mapping subset of Figure 6, it is possible to see that several conditions govern the mapping of "atomic" ExtendedActivity. These vary depending on the type of resource(s) supporting the activity: (1) if a Human resource is responsible for the realization of the Extended Activity, it is mapped to a UserTask; (2) if an IT resource is responsible for the realization of the Extended Activity, it is mapped to a ServiceTask. Full mapping is available in Bazoun et al., (2013).

EA*	Condition		BPMN2.0	
Model			Definitions	
Process			Pool, Process, and Participant	
Extended Activity	Structural		Activity	Sub Process
	Atomic	It is supported by Human		UserTask
		It is supported by IT (no human interaction)		ServiceTask
LogicalOperator	DivergingOr		Gateway	Diverging Exclusive Gateway
	ConvergingOr			Converging Exclusive Gateway
	DivergingAnd			Parallel Gateway
	ConvergingAnd			Parallel gateway
Resource	Material		Data Object	
	Human	Responsible for	Lane	
		Participates in	Resource (added to the list of resources of a task)	
	IT	Responsible for	Lane	
		Participates in	Resource (added to the list of resources of a task)	

Figure 6: EA* and BPMN 2.0 Mapping Subset.

For the implementation, ATL was elected. It is a largely used language to implement MDA based tools, having a specific development toolkit plug-in available in open source (Eclipse Modelling Project - http://www.eclipse.org/modeling/) (Jouault & Kurtev, 2007). The ATL rule mechanism provides developers with a convenient means to specify the way target model elements must be generated from source model elements. For this purpose, a matched rule enables to specify:

- Which source element must be matched;
- The number and the type of the generated target model elements;
- The way these target model elements must be initialized from the matched source elements.

```
rule ExtendedActigramToDefinition{
  from s: EA!EaModel
  to a: BPMN!Definitions (
    id <- s.name,
    targetNamespace <-'www.jboss.org/drools',
    expressionLanguage <- 'www.mvel.org/2.0',
    typeLanguage <- 'www.java.com/javaTypes',
    rootElements <-
       thisModule.ProcessToProcess(s.process)
  )
}
```

Figure 7: Matched Rule.

Figure 7 contains an example of a "matched rule" to

transform an ExtendedActigram element to a Definition element. It is identified by its name, and is composed of two mandatory (the "*from"* and the "*to"* parts) and an optional (the *do* parts) sections. The "from" part represents the concept to be mapped from the Extended Actigram Star meta-model which is ExtendedActigram and the "to" part represents its corresponding concept in BPMN meta-model which is Definitions. The declarative block of the "to" part is used for assigning values to attributes.

```
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(…),
    laneSets <- thisModule.laneSet
  )
  do {
       thisModule.bpmnProcess <- a;
       thisModule.bpmnProcess.flowElements   <-
thisModule.bpmnProcess.flowElements.union(this
Module.bpmnFlowElements);
       thisModule.bpmnProcessRef <-a;
       thisModule.eaStarProcessRef <- s;
       …}
}
```

Figure 8: Lazy Rule.

In our implementation, the ATL mechanism of "lazy rules" has also been used, called from match rules. They have the same structure but are applied only on the element passed as a parameter. "thisModule. ProcessToProcess(s.process)" from Figure 7 calls the lazy rule "ProcessToProcess" on the element "s.process". It is evident that the rule included in Figure 8 calls other rules, yet both "to" and "do" parts have been shortened ("…") due to space constraints.

Figure 9 illustrates graphically how the results of applying the transformation from an EA* model to a BPMN model looks like.

6 CONCLUDING REMARKS, DISCUSSION AND FUTURE WORK

Based on model transformations, MDSEA has the challenge to unify every step of the enterprise (or virtual enterprise) servitization. It answers to the requirements of service system re(engineering) while maintaining portability and interoperability, enabling value co-creation.

The proposed methodology and transformations framework brings bi-dimensionality and automation to the servitization system. Having MDA/MDI as the enabling technology, vertical and horizontal dimensions are supported implementing a comprehensive transformations architecture that recognizes semantics as an important area for models alignment. Following the architecture and applying the mappings presented, some ATL transformations have been implemented and executed in the frame of an industrial scenario.

From a user's perspective, feedback has been received stating that modelling actually facilitates the requirement collection, and the model driven approach allows getting to code level more efficiently and robustly. Re-engineering is no longer seen as a problem.

From a developer's perspective, the mappings are time-consuming processes that once defined can be executed any number of times achieving the same results. Nevertheless, despite its robustness, the ATL technology represents models as meta-models and relates their properties through static rules. This means that each time that it is necessary to change relations between models, manual codification is required to recreate them. This behavior is a direct consequence of dynamism flaw.

In this context, the application of communication mediation ontologies in the line of the work presented by Sarraipa et al., (2010) are considered relevant for the future work, and positive influencers for the success of MDSEA-based transformations. As parseable knowledge repositories, properly instantiated with domain data, they can support automation by enabling intelligent services to work on top of them and facilitate the intervention of the Human actor (both user and developer).

Another line of future work concerns the simulation of models to evaluate the behaviour of the system regarding time and identify desired or undesired behaviour, including the respect of causal relations. Another step proposed for this research is to add one new transformation of BPMN models to DEVS models in the goal of running simulations. Some on-going works have started reusing, in the context of Service process modelling and simulation, already existing matching between Workflow and DEVS (Zacharewicz et al., 2008).

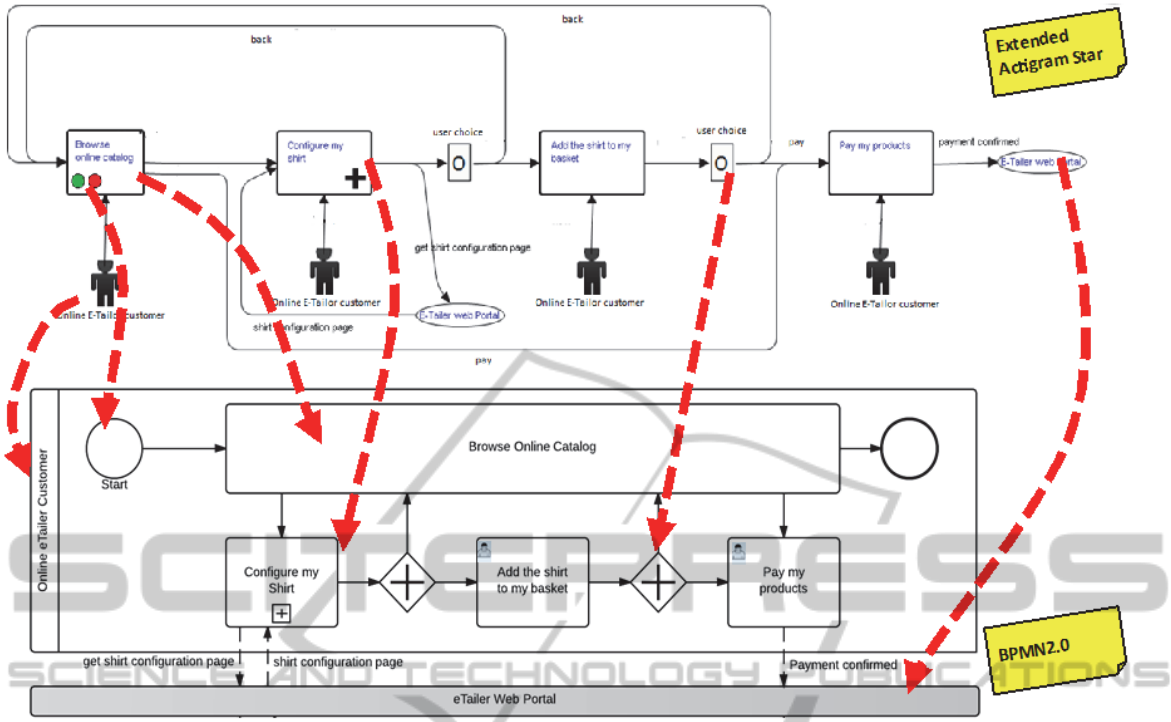

Figure 9: Graphical example of mapping and transformation of models.

ACKNOWLEDGEMENTS

Authors would like to acknowledge the European funded Project MSEE (FP7 284860) that supported the development of various ideas, concepts and use case presented in this paper.

REFERENCES

Agostinho, C., Černý, J. & Jardim-goncalves, R. (2012) MDA-Based Interoperability Establishment Using Language Independent Information Models. In: M. van Sinderen, P. Johnson, X. Xu, & G. Doumeingts eds. 4th International IFIP *Working Conference on Enterprise Interoperability* (IWEI 2012). Harbin, China, Springer, pp.146–160.

Agostinho, C., Jardim-Goncalves, R., Sarraipa, J., Lampathaki, F., Koussouris, S., Charalabidis, Y., Psarras, J., Assogna, P., Missikoff, M., Popplewell, K., Silva, E. & Ferreira, J. (2012) Deliverable D2.4: EISB *Models & Tools Report*. ENSEMBLE CSA Project (FP7-ICT-257548).

Athena IP (2006) ATHENA Interoperability Framework (AIF) (Internet). Available from: < http:// athena.modelbased.net > (26 Nov 2013).

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

Bazoun, H., Zacharewicz, G., Ducq, Y. & Boye, H. (2013) Transformation of Extended Actigram Star to BPMN2.0 in the frame of Model Driven Service Engineering Architecture. In: *Symposium on Theory of Modeling and Simulation* (TMS/DEVS 2013). San Diego, CA, USA.

Bézivin, J. (2005) Model Driven Engineering: An Emerging Technical Space. In: Generative and *Transformational Techniques in Software Engineering, International Summer School* (GTTSE 2005). Braga, Portugal.

Chen, D. & Doumeingts, G. (1996) The GRAI-GIM reference model, architecture and methodology. In: P. Bernus, L. Nemes, & T. J. Williams eds. *Architectures for Enterprise Integration. London*, UK, Chapman & Hall.

Chesbrough, H. & Spohrer, J. (2006) A research manifesto for services science. *Communications of the ACM*, 49 (7), p.35.

Doumeingts, G., Vallespir, B. & Chen, D. (2006) GRAI GridDecisional Modelling. In: P. Bernus, K. Mertins, & G. Schmidt eds. *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, pp.321–346.

Ducq, Y., Chen, D. & Alix, T. (2012) Principles of Servitization and Definition of an Architecture for Model Driven Service System Engineering. In: 4th

International IFIP *Working Conference on Einterprise Interoperability* (IWEI 2012). Harbin, China, Springer.

Ducq, Y., Doumeingts, G., Lieu, C., Chen, D., Alix, T. & Zacharewicz, G. (2012) Deliverable D11.2: Service concepts, models and method: Model Driven Service Engineering (M12 issue). *MSEE IP Project* (FP7 FoF-ICT 284860).

Estefan, J. A. (2007) Survey of Model-Based Systems Engineering (MBSE) *Methodologies*. INCOSE. From: <www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf> (Accessed 7 September 2011).

Frankel, D. (2003) Model Driven Architecture – *Applying MDA to Enterprise Computing*. 1st edit. OMG Press.

Garschhammer, M., Hauck, R., Hegering, H.-G., Kempter, B., Radisic, I., Rolle, H., Schmidt, H., Langer, M. & Nerb, M. (2001) Towards generic service management concepts a service model based approach. In: 2001 IEEE/IFIP *International Symposium on Integrated Network Management Proceedings. pp.719–732*.

ISO TC184/SC5 (2000) Industrial Automation Systems—Requirements for Enterprise-reference Architectures and Methodologies (ISO 15704:2000).

Jardim-Goncalves, R., Agostinho, C. & Steiger-Garcao, A. (2012) A reference model for sustainable interoperability in networked enterprises: towards the foundation of EI science base. *International Journal of Computer Integrated Manufacturing, 25 (10)*, pp.855–873.

Jardim-Goncalves, R., Grilo, A., Agostinho, C., Lampathaki, F. & Charalabidis, Y. (2013) Systematisation of Interoperability Body of Knowledge: the foundation for Enterprise Interoperability as a science. E*nterprise Information Systems, 7 (1)*, pp.7–32.

Jouault, F. & Kurtev, I. (2007) On the interoperability of model-to-model transformation languages. *Science of Computer Programming, 68*, pp.114–137.

Kohlborn, T., Korthaus, A. & Rosemann, M. (2009) Business and software lifecycle management. In: 2009 *Enterprise Distributed Object Computing Conference* (EDOC '09).

OMG (2011a) Business Process Model and Notation (BPMN) - version 2.0 (formal/2011-01-03). Available from: <http://www.omg.org/spec/BPMN/2.0/PDF/>.

OMG (2003) MDA Guide Version 1.0.1 (omg/2003-06-01). Object Management Group. Available from: <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf> [Accessed 7 September 2011].

OMG (2011b) OMG Unified Modeling LanguageTM (OMG UML), Infrastructure - version 2.4.1. From <www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/>

Sarraipa, J., Jardim-Goncalves, R. & Steiger-Garcao, A. (2010) MENTOR: an enabler for interoperable intelligent systems. *International Journal of General Systems, 39 (5)*, pp.557–573.

Selic, B. (2003) The pragmatics of model-driven development. IEEE Software, 20 (5), pp.19–25.

Spohrer, J., Maglio, P., Bailey, J. & Gruhl, D. (2007) Steps Toward a Science of Service Systems. Computer, 40 (1), pp.71–77.

Thoben, K., Eschenbächer, J. & Jagdev, H. (2001) Extended Products: Evolving Traditional Product Concepts. In: *7th International Conference on Concurrent Enterprising. Bremen*, Germany.

Truyen, F. (2006) (White Paper) The Fast Guide to Model Driven Architecture: The Basics of Model Driven Architecture. Available from: <http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf>.

Vandermerwe, S. & Rada, J. (1988) Servitization of business: adding value by adding services. *European Management Journal, 6 (4)*, pp.314–324.

Zacharewicz, G., Frydman, C. & Giambiasi, N. (2008) G-DEVS/HLA *Environment for Distributed Simulations of Workflows*. SIMULATION, 84 (5), pp.197–213.

Zdravkovic, M., Panetto, H. & Trajanovic, M. (2013) Semantic Interoperability for Dynamic Product-Service. In: *International Conference on Information Systems and Technology* (ICIST 2013). Kopaonik, Serbia.