Transformation of extended actigram star to BPMN2.0 and simulation model in the frame of model driven service engineering architecture

Hassan Bazoun, Gregory Zacharewicz, Yves Ducq, Hadrien Boyer

Transformation of Extended Actigram Star to BPMN2.0 and Simulation Model in the Frame of Model Driven Service Engineering Architecture

Conference Paper · April 2013

CITATIONS
13

READS
286

4 authors, including:

Hassan Bazoun
Hardis Group
11 PUBLICATIONS **188** CITATIONS
SEE PROFILE

Gregory Zacharewicz
IMT Mines Alès
210 PUBLICATIONS **1,527** CITATIONS
SEE PROFILE

Yves Ducq
University of Bordeaux
177 PUBLICATIONS **1,792** CITATIONS
SEE PROFILE

Transformation of Extended Actigram Star to BPMN2.0 in the frame of Model Driven Service Engineering Architecture

Hassan Bazoun, Gregory Zacharewicz, Yves Ducq, Hadrien Boye
Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France.
Hardis / Hardis Conseil, 3 rue Guglielmo Marconi, 44800 Saint Herblain, France
hassan.bazoun@ims-bordeaux.fr, gregory.zacharewicz@ims-bordeaux.fr, yves.ducq@ims-bordeaux.fr, hadrien.boye@hardis.fr

Keywords: Interoperability, service engineering, model transformation, Extended Actigram Star, BPMN, G-DEVS Simulation.

Abstract

Cooperation between different enterprises to provide product and related services has become a must in order to set up win-win alliances and benefit better from market opportunities. This evolution has encountered several problems, like interoperability when trying to exchange data between heterogeneous systems. This paper shows how a model-driven approach can be an answer to service system implementation and to interoperability problems. In particular it details the necessity to provide transformation mechanisms from conceptual description here Extended Actigram Star (EA*) models to more technical models such as BPMN 2.0 models. At the end the paper describes a last transformation to G-DEVS simulation models in order to validate, thanks to simulation, some behavioral properties of the BPMN model before going to implementation.

1. INTRODUCTION

Traditional manufacturing enterprise, in Europe and around the world, will progressively migrate from traditional product-centric business to product-based service-oriented virtual enterprise and ecosystems [1]. Therefore, these companies have to cooperate in one or several virtual enterprises, considered as service systems to support the service life cycle. In order to manage this transition from product oriented toward service oriented business, the various virtual manufacturing enterprises should be modeled, designed, implemented, tested and managed along its entire. However, to properly implement the service system, it is good to separate the user position from technical point of view, with a model driven approach.

This paper presents some preliminary results of a research work performed in the frame of the FP7 MSEE (Manufacturing Service Ecosystem) Integrated Project [2]. One of the results of MSEE is the development of a Model Driven Service Engineering Architecture (MDSEA) in which transformation of models is included. Also this paper will show the work done in the domain of model transformation between two modeling languages used in the service architecture of MSEE

The next part will present the principles of MDSEA, justify the chosen modeling languages at each modeling level and the need of model transformation between modeling levels and then between modeling languages. Then, a literature review on model transformation will be done. After, the mechanisms of model transformation between Extended Actigram Star and BPMN 2.0 will be presented in details. Then, a final transformation to G-DEVS simulation model is presented emphasizing the usage of simulation in model validation and service testing. Finally the perspectives of this work will be proposed at the end of the paper.

2. MDSEA

The objective of a model driven approach is to separate between the business and technical point of view in product-service systems. An engineering architecture specifies a framework (i.e. a conceptual structure) for engineering activities, which provides a set of guidelines for structuring the specifications organized with various abstraction levels.

The Model Driven Service Engineering Architecture (MDSEA) is inspired from MDA [3]/MDI [4] (Model Driven Architecture/ Model Driven Interoperability). MDA defines three modeling levels and specifies the goals that must be followed at each level but without mentioning how to model or which modeling language to be used as it is proposed in MDSEA. The MDI approach is more detailed but focuses only on IT aspects. On the other hand, MDSEA supports the need for modeling the three types of components (IT, Organization/Human and Physical Means) which form a "service system". In this sense, it is therefore considered as an adaptation 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 modeling based on three abstraction levels: BSM, TIM and TSM as well as the dedicated modelling languages at each level.

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 from 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 to understand a problem, but also to bridge the gap between domain experts and development experts who will build the service system (adapted from Miller, et al., 2003). 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. TIM levels represent the same system but with more detailed specifications. 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 technology related details used for implementation. At TIM level, the detailed specification of a service system's components will be elaborated with respect to IT, Organization/Human and Physical means involved within the service production. TIM can be derived partially from BSM models.

2.3. Technology Specific Model (TSM)

TSM enhance 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 components or material handling) and calls to Human resources related tasks/operations.

2.4. Conclusion

The proposed MDSEA aims to provide and integrate a set of modeling languages at the different abstraction levels, to support service system design and implementation. The desired service system will be first specified and represented globally from a business user's point of view at the lower level of the global modeling. Then detailed modeling and specifications will allow determining the three types of components (IT, Organization/Human, Physical means) that are necessary to realize the service system. Finally, related descriptions and specifications will be delivered with sufficient details to build the design service system.

So, based on these modeling levels, it is proposed to associate relevant modeling languages at each level. At the business level, the modeling languages must be very simple, powerful and understandable by business oriented users. Moreover, these languages must cover the process modeling and decision modeling in a coherent way.

As for process modeling, a lot of languages exist and Extended Actigrams Star (EA*), derived from IDEF0 was chosen to model processes at BSM level due to its generic modeling of resources: machine, human and IT. The hierarchical approach of EA* was also a reason of this choice.

At the TIM level, OMG BPMN 2.0 (Business Process Modeling Notations) [5] was chosen in particular because this language offers a large set of detailed modeling constructs, including IT aspects and benefits from the interoperability of many BPM IT platforms allowing the deployment and automatic execution of BPMN processes.
However, because the languages are not the same, it is necessary to transform EA* models into BPMN 2.0 models in order to obtain business process models at TIM level based on those previously modeled at BSM level. As a consequence, the next part will present a state of the art in model transformation and then the transformation mechanisms and rules between EA* and BPMN 2.0 will be presented.

3. STATE OF THE ART

Business process is a collection of related, structured activities or tasks that produce a specific service or product for a particular customer(s). It is "the structure by which an organization does what is necessary, to produce values for its customers" [6].

The standards of process modeling are gaining more and more importance, which gave rise to several process modeling languages and tools to enhance the representation of enterprise processes. One of these languages is the GRAI Extended Actigram [7] which intends to capture business process models at a high semantic level, independently from any technological or detailed specifications. It is an extension of IDEF0 [8] Actigram language. Several attempts tried to bridge the gap between GRAI Extended Actigram and other process modeling languages, such as BPMN.

ASICOM [9] 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 [10] were tested and evaluated.

The ASICOM team has encountered several problems during his 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 doesn't have an official MOF metamodel, but several metamodels developed in the frame of academic researches and projects. In addition, the specification of Grai Extended Actigram is not sufficiently formal to allow the transformation into other formalisms.

This paper presents an improved version of the Grai Extended Actigram language called Extended Actigram Star (EA*), developed as an answer to previous issues encountered with GRAI extended actigram language regarding its interoperability. In addition, the paper highlights the transformation from the developed Extended Actigram Star models to BPMN2.0 models.

4. TRANSFORMATION'S PRINCIPLES

Model transformation provides means to produce target models from different source models [11]. For this purpose, it permits the definition of how source model elements must be matched in order to initialize target model elements.

This section introduces the main principles of transforming an Extended Actigram Star Model into a BPMN model, including the proposition of the Extended Actigram Star language, the transformation architecture specific to our domain of study, the mapping of Extended Actigram star concepts to BPMN2.0 concepts, and the transformation language used to implement this mapping.

4.1. Extended Actigram Star

Extended Actigram Star (EA*) relies on previous work developed in the frame of the GRAI Methodology [24], which defines "GRAI Extended Actigram" as a process modeling language, among other graphical formalism, for enterprise modeling and "decision centric" analysis.

The goal of Extended Actigram Star is to:

- Provide a common modeling notation comprehensible by business users for business process description.
- Reduce the gap between the ideation and the design of business process (by its simple and accessible syntax).
- Facilitates the transformation of business process models toward other structured modeling languages offering more detailed constructs (e.g. BPMN2.0).

4.1.1. Conceptual Model

The conceptual model is formed of several regrouping levels to generalize concepts and to factor out details.

Extended Actigram Star elements are divided into three sub packages: **Root package** containing the root element of the Extended Actigram star Language (Model)**, General Elements package** that reduce and factor out details, and **Core Elements package** that contains every concrete element that has a corresponding graphical representation defined by this language.

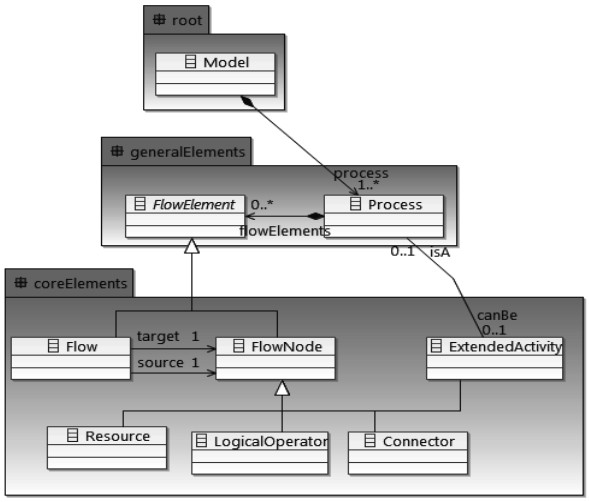

Figure 1 EA* conceptual model

All Extended Actigram Star elements inherit from the BaseElement class three common attributes: **id**, **name**, and **code**.

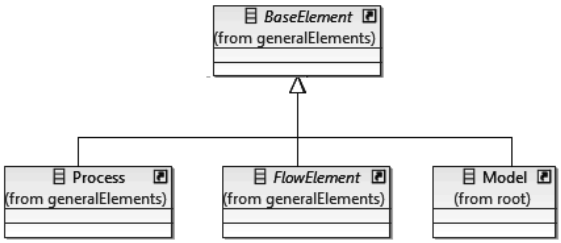

Figure 2 BaseElement

Extended Actigram Star diagram is a representation of a business process (the subject to be modeled). A **Process** is composed of FlowElement(s), which is an abstract representation of all elements constituting the diagram.

A FlowElement can be either a **Flow,** used to link FlowNodes, or a **FlowNode** that is an abstract representation of the diagram's elements that are connected together by means of flows.

A FlowNode is a supper class of four other classes:

- **ExtendedActivity:** this represents the functional unit of a process. An Extended activity can be broken down into several activities. In such case, it is called a 'Structured Activity'. An activity that has not been broken down will be called an "Atomic Activity'.
- **Resource:** an abstract concept representing resources used by a process to support one or several activities. It can be of three types: human, material, and IT.

- **Connector:** used to represent the origin or the destination of a flow when the origin or the destination is outside the current diagram. Possible roles are: (process, internal, and external) connector.
  A ProcessConnector class has a reference pointing to a process outside the current diagram.
- **Logical Operator**: this represents a convergence or a divergence of multiple flows. There are four different kinds of logical operators: ConvergingAnd, DivergingAnd, ConvergingOr, and DivergingOr.

4.2. Transformation architecture

The objective is to transform an Extended Actigram Star (EA*) model into a BPMN 2.0 target Model. One of the most used transformation techniques is the "Metamodel Approach" [3]. Figure 3 is a particularization of the "Metamodel" approach to the EA* to BPMN 2.0 context.

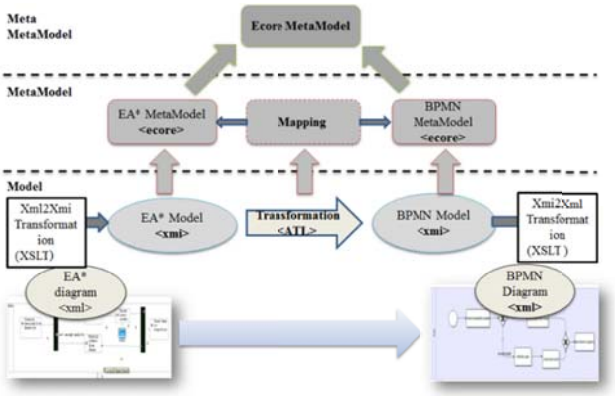

Figure 3 Transformation architecture of EA* to BPMN

The first step consists of identifying the source and target metamodels (respectively Extended Actigram Star metamodel and BPMN2.0 metamodel), and the languages used for the model representation. The Ecore metamodel [13] is used as a meta-metamodel which defines the structures of the Extended Actigram star and BPMN metamodels. Both EA* and BPMN2.0 metamodels are defined in ecore. In addition XML Metadata Interchange (XMI) [14] is used to save source and target models. After defining the metamodels and the model's description languages, a mapping between metamodels is built. This mapping can be characterized by a table defining constructs' matching. Later, the mapping rules described in the table will be implemented using Atlas transformation Language (ATL) [15].

The Eclipse plugin "BPMN modeler" [16] is used to visualize and validate BPMN target models. It requires a specific xml format [5] with graphical elements' definition. To address this issue, it relies on XSLT [17] transformation sheet which will transform the BPMN models (xmi format) resulting from the ATL transformation into xml format that conforms to the BPMN modeler requirements.

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. Table 1 summarizes the mapping of EA* concepts to BPMN concepts. The mapping is accompanied with conditions which govern relations between concepts.

4.3.1. ExtendedActivity

In contrast to "structural" ExtendedActivity, several conditions govern the mapping of "atomic" ExtendedActivity. These conditions vary depending on the type of Resource(s) supporting the ExtendedActivity.

Condition1: A human interaction is needed to perform the ExtendedActivity, and thus it is supported by a Human resource (it might be supported by several resources of different types, but there is at least a Human resource). The atomic Extended Activity is mapped to a UserTask.

Condition2: no human interaction is needed and it is supported by an IT resource. In this case the Extended Activity is mapped to a ServiceTask.

Condition3: it is only supported by Material resources.
In this case the ExtendedActivity is mapped to a Task.

4.3.1. "Control" Flow

The mapping of a Flow of type "Control" is dependent on the type of source and target connected by the flow.

Condition1: Source is an ExternalConnector or InternalConnector and target is an "atomic" ExtendedActivity. It is mapped to a MessageFlow.

Condition2: 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 SequenceFlow.

Condition3: Source is a ProcessConnector or ExtendedActivity. It is a "1 to n" relation, the "Control" Flow is mapped to a DataObject and two Associations.

4.3.2. "OutputInput" Flow

The mapping of an "OutputInput" Flow is dependent on the type of source and target connected by the flow.

Condition1: Source is an ExternalConnector or InternalConnector and Target is an atomic Extended Activity. "OutputInput" Flow is mapped to MessageFlow.

Condition2: Source is an ExternalConnector or InternalConnector and Target is not an atomic Extended

Activity. This case is a "1 to n" relation, in which the "OutputInput" Flow is mapped to a combination of MessageFlow catching MessageEvent and SequenceFlow.

Condition3: Source/target are ProcessConnectors, Logical-Operators or ExtendedActivities. It is SequenceFlow.

Condition4: 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 "OutputInput"Flow is mapped to a combination of MessageFlow, MessageEvent, and a SequenceFlow.

Condition5: Source is an atomic ExtendedActivity and target is an ExternalConnector or InternalConnector. In this case it is mapped to a MessageFlow.

4.3.1. "Support flow"

The mapping of "Support" Flow depends on the source.

Condition1: Source is a Material resource. In this case it is mapped to an Association.

Condition2: Source is not a Material resource. In this case it is not mapped. (See mapping of resources in Table 1).

4.4. Model Transformation Language

This section shortly presents the model transformation language used based on ATL [15] to implement the mapping of concepts (Table 1).

ATL is a model transformation language specified as both a metamodel and a textual concrete syntax. In the field of Model-Driven Engineering (MDE), ATL provides developers with a mean to specify the way to produce a number of target models from a set of source models.

The ATL language is a hybrid of declarative and imperative programming. The preferred style of transformation writing is the declarative one: it enables to simply express mappings between the source and target model elements. However, ATL also provides imperative constructs in order to ease the specification of mappings that can hardly be expressed declaratively.

An ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the target models elements.

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
		It is only supported by material		Task
LogicalOperator	DivergingOr		Gateway	Diverging Exclusive Gateway
	ConvergingOr			Converging Exclusive Gateway
	DivergingAnd			Parallel Gateway
	ConvergingAnd			Parallel gateway
Resource	Material		Data Object	
	Human		Performer in a "UserTask"	
	IT		Resource (added to the list of resources of a task)	
Flow	Control	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	
	OutputInput	If the source is an ExternalConnector or InternalConnector (and target is an atomic Extended Activity)	MessageFlow	
		If the source is an ExternalConnector or InternalConnector (and target is a structural Extended Activity or LogicalOperator)	Catching Message Event, Message Flow, and Sequence Flow	
		If the source is a ProcessConnector, ExtendedActivity, or LogicalOperator (and target is ProcessConnector, ExtendedActivity, or LogicalOperator)	SequenceFlow	
		If the source is a structural ExtendedActivity or LogicalOperator (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 ExternalConnector or InternalConnector)	MessageFlow	
	Support	If source is a Material resource	Association	
Connectors	External		Participant (Pool)	
	ProcessConnector		Call Activity	

	InternalConnector	Participant(Pool) (Black BOX)

Table1 mapping of concepts

4.4.1. Matched rules

The ATL matched rule mechanism provides developers a convenient mean to specify the way target model elements must be generated from source model elements.

```
rule Material2DataObject {
from s: EA!Resource (s.oclIsTypeOf(EA!Material) )
to k: BPMN!DataObject (  id <- s.id,    name <- s.name )}
```

Figure 4 Matched rule

The Material2DataObject rule transforms every Material element into a DataObject element

4.4.2. Lazy rules

Lazy rules are like matched rules, but are only applied when called by another rule.

```
lazy rule ProcessToProcess {
from s: EA!Process ( s.oclIsTypeOf(EA!Process) )
to k: BPMN!Process (
    id ←s.id,
   name ← s.name,
   flowElements ← s.flowElements )
do{
   thisModule.processRef ← k;
   thisModule.collaborations.participants ←
   thisModule.collaborations.participants.
   append(thisModule.ProcessToParticpent(k));}}
```

Figure 5 Lazy rule

The Process2Process rule will transform a specific EA* process element into a BPMN process element.

4.4.3. Called rules

Called rules explicitly generate target model elements from imperative code. Except for entrypoint called rule that must be explicitly called from an ATL imperative block.

```
entrypoint rule CreateCollaboration() {
to t: BPMN!Collaboration (  name ← 'collaboration')
do{   thisModule.collaborations ← t ; }}
```

Figure 6 Called rule

The CreateCollaboration rule is implicitly invoked at the beginning of the transformation execution. It creates a Collaboration element with the name “collaboration”.

4.5. Example

Figure 7 is an example of a transformation from EA* diagram to BPMN diagram. The EA* diagram (upper diagram) is modeled using an EA* graphical editor. The diagram is a representation of an order process within an enterprise, in which a check on availability and credits is performed before fulfilling the order or rejecting it.

The second diagram is a BPMN2.0 diagram viewed using the BPMN modeler. Both diagrams were created using the SLMToolBox which is a modeling tool developed in the frame of the MSEE project.

5. FROM BPMN TO SIMULATION MODELS

The final step is leaded by the fact that BPMN is not ready for execution it misses the temporal dimension.
Several works were proposed in literature to transform BPMN models to simulation models. For instance in [18] and [19] the authors proposed a new model driven solution to generate simulation models dedicated to distributed environment context. Moreover, in [20], the authors have proposed a method to transform the (semi-formal) Workflow graphical models into (formal) G-DEVS (Generalized-DEVS) coupled models by connecting G-DEVS atomic models representing the Workflow basic components. Nevertheless the Workflow components were not mature and not formally described leading to many interpretations in the source model. Also in [21] a first step to transform BPMN to DEVS has been proposed, the authors have proposed matching for major BPMN components to DEVS models. Nevertheless not all components of BPMN 2.0 were detailed and matched into DEVS models. These works have been extended for transforming BPMN 2.0 to G-DEVS.

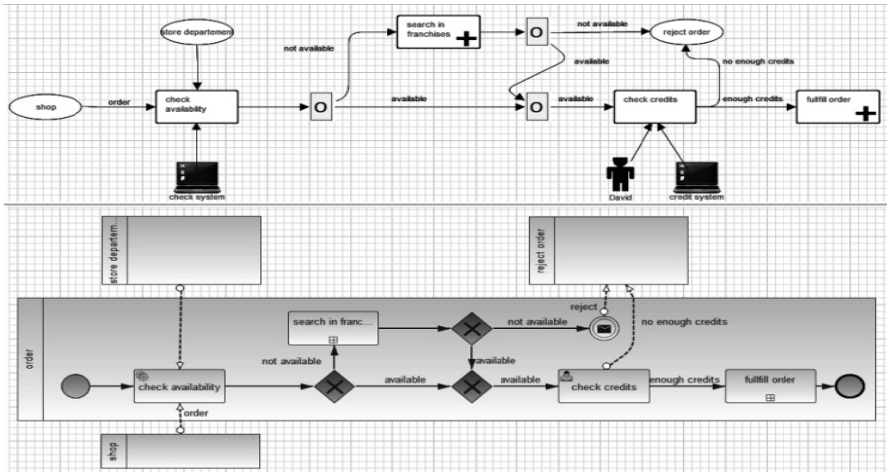

Figure 7 EA* model to PBMN2.0 model example

The choice of G-DEVS as a formal modeling and simulation language is based on several reasons. First, a G-DEVS model takes advantage of formal properties; it describes clearly time information. Second it can be simulated with the abstract simulator specified in [22] clearing ambiguities that can occur with some simulation languages due to different implementations. Requirements for modeling and simulating service process workflows, were based on the capacity to capture all characteristics of service processing according to latest specification of BPMN 2.0. Services' characteristics evolve during execution time, and thus it can be considered as state variables with values that are changing during processing within Workflow. As a result, the goal was to capture important data and to follow up this information. In more detail, they need to be described by several attributes including their references, routes, composition and progress status. This complex state is evolving during progress. To address these objectives, G-DEVS has been chosen as particularly convenient since it allows the concept of multiple attributes (values) carried in one event. In our environment the services are described by multiple attributes of a G-DEVS event. In addition the G-DEVS coupled models allow us to easily compose a BPMN workflow by coupling: service processing, tasks, resources, and routing sequences; the behavior of each component is described in G-DEVS atomic model.

The authors have developed, in particular, a G-DEVS model of the BPMN 2.0 "Task" component [23], according to OMG latest specification.. This model considers all possible inputs in the BPMN standard. It clearly distinguishes the message flow and sequence flow. It also adds the compensation events, timer, and default scenario to be taken into account by the model. The G-DEVS atomic model in Figure 6 was developed with LSIS_DME [20].

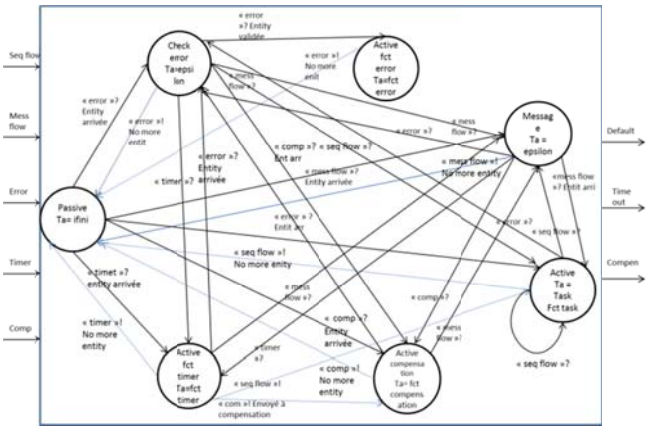

Figure 8 G-DEVS Model of BPMN Task

In addition G-DEVS blocks from [20] have been reused for queue management and resource allocation that can help to reveal by simulation the problem of bad resource allocation or wrong dimensioning of services building.

Simulations were executed to in the goal of validate some execution scenario provided by the MSEE project. In particular some KPI where identified and targeted to be reached during the service generation process, such as time, cost and some more qualitative like satisfaction of user based on service coverage. Also the simulation runs has revealed some limitations, which were mostly based on the fact that quantitative data were not available to be collected at specification and interviews with expert steps. Authors stress the fact that the goal is not directed by simulation performance but more by interoperability that can be useful for plugin the simulation with other execution components including data bases and ERP components to launch mixed simulation and interaction with human in the loop.

6. PERSPECTIVES

The idea is to push further the interoperability of the G-DEVS BPMN Environment to be distributed. A key to these requirements is to use the High Level Architecture (HLA) standard as a common way to share synchronized data between them. Authors presented in [18] that models can be run from several distributed components and places. Thanks to the capability of G-DEVS models to be integrated as create HLA federates introduced in [20], the interoperability will be facilitated. This desired capability matches with the distribution requirements of actual industrial service processes. Indeed, actual real industrial processes required the use of human decision and multiple tools that interact with the process at the different steps of the service definition and generation. The different software tools are heterogeneous and need to cooperate. Thus, the authors will propose to use SLMToolBox as the editor/simulator engine of a distributed BPMN Workflow environment and to generalize the HLA compliance to the whole BPMN modeling environment by adding other federates to the federation in order to define a Distributed Workflow Reference Model.

7. CONCLUSION

This paper expressed the problem of interoperability for collaborative enterprises that tend to exchange services. It introduced the model driven service engineering architecture (MDSEA) as an extension of the MDA/MDI approaches that can support solving this interoperability problem. The use of model based approach was exposed. The first modeling language introduced is GRAI Extended Actigram to model at conceptual level and then BPMN language has been recalled for tackling model technical details. Apart from syntactical transformation, that can be almost fully automatized, the need of a well-defined model transformation in the frame of MDSEA avoiding semantic loss has been identified.

Then, transformation architecture was proposed; it governs the transformation attempt which is based on the "metamodel approach" transformation architecture. The mapping of concepts has been detailed using a table representing the links and relations created between concepts of both conceptual models. And after, ATL rules were introduced in order to implement the maooing table followed by a concrete example of the transformation from Extended Actigram Star to BPMN.

At the end a final transformation to G-DEVS models is introduced. The idea was to reuse and complete already proposed transformations to permit an almost entirely linked transformation from top level to simulation in order to validate behavioral properties of the models. The last outstanding and remaining issue at the end is based on the lack of data for time consideration on the service building and delivery.

Acknowledgement

This work has been partially supported by the FP7 Project ID 284860 MSEE project.

References

[1] Thoben, K.-D., Jagdev, H., Eschenbächer, J. (2001) Extended Products: evolving traditional product concepts. In: K.-D. Thoben, F. Weber and K.S. Pawar (ed.) Proceedings of the 7th International Conference on Concurrent Enterprising: "*Engineering the Knowledge Economy through Co-operation*" Bremen, Germany, June 2001.

[2] FP7 – FoF-ICT-2011.7.3 – "*Manufacturing SErvice Ecosystem Project- Annex 1 description of work*" – July 29th 2011

[3] OMG, "*MDA Guide Version 1.0*" document number: omg/2003-05-01.

[4] JP. Bourey, R. Grangel, G. Doumeingts, A. Berre. Deliverable DTG2.3 from the INTEROP project. "*Report on Model Driven Interoperability*"

[5] OMG, "*Business Process Model and Notation (BPMN) version 2.0*" document number: formal/2011-01-03. [6] Thomas Davenport (1993). "*Process Innovation: Reengineering work through information technology*". Harvard Business School Press, Boston

[7] G. Berio, Project UEML, WP3, Deliverable D3.3, "*Requirements Analysis: initial core constructs and architecture*", Annexe2.2, Tech. rep. (2003).

[8]http://www.idef.com/IDEF0.htm

[9] "*Etat des lieux critique sur MDI*" ASICOM Liverable3.1

[10] "*Design and Development for Transformation from GRAI to BPMN module and Semantic Annotations for BPMN module*" ASICOM FUI-DGE Technical report

[11]http://www.emn.fr/z-info/atlanmod/index.php/Model_Transformation

[12] OMG, *"Object Constraint Language"* document number: formal/2006-05-01

[13] Ken McNeill "*How to extend the Eclipse Ecore metamodel*"
http://www.ibm.com/developerworks/library/os-eclipse-emfmetamodel/index.html

[14] OMG, "*XML Metadata Interchange (XMI)*", document number: ad/2001-06-12 http://xml.coverpages.org/OMG-XMI-ad20010612.pdf

[15] "*ATL/User Guide – The ATL Language*" http://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language

[16] http://eclipse.org/bpmn2-modeler/

[17] "*XSLT Tutorial*" http://www.w3schools.com/xsl/

[18] Bocciarelli P, Pieroni A., Gianni D., D'Ambrogio A. (2012) *"A model-driven method for building distributed simulation systems from business process models"*. Winter Simulation Conference 2012 p. 227

[19] Bocciarelli P., D'Ambrogio A. (2012) *"Automated performance analysis of business processes"*. p. 10, SCS SpringSim (TMS-DEVS)
[20] Zacharewicz, G.; Frydman, C.; Giambiasi, N., *"G-DEVS/HLA Environment for Distributed Simulations of Workflows, Simulation",* Volume 84, issue 5, (May 2008), p. 197-213.
[21] Deniz Çetinkaya, Alexander Verbraeck, Mamadou D Seck (2012) *"Model Transformation from BPMN to DEVS in the MDD4MS Framework"*, p. 304-309. SCS SpringSim (TMS-DEVS).
[22] Zeigler, B.P., H. Praehofer and T.G. Kim. 2000. *"Theory of Modeling and Simulation"*. 2nd Edition. Academic Press: New York, USA.
[23] Kazma, R. (2012) *"Transformation of BPMN to G-DEVS modeling for simulation"*, Master Thesis Report, University of Bordeaux 1.
[24] Doumeingts G., Ducq Y. (2001) "Enterprise Modelling techniques to improve efficiency of enterprises"
in International Journal of Production Planning and Control - Taylor & Francis – Vol. 12, number 2, Mars 2001, pp 146-163