A transformation of YAWL to AToMPM Modeling Language

Srinivasan Balakrishnan
(Srinivasan.Balakrishnan@student.uantwerpen.be)[†]

Master of Computer Science, University of Antwerpen

September 2, 2015

Abstract

Modeling languages have been used for software development and industrial applications. YAWL(Yet Another Workflow Language) is one of the leading design pattern. Many business processes using this simulation environment to manage, analyze, flowchart their needs. The interesting thing is to design one modeling language in another. AtoMPM(A Tool for Multi-Paradigm Modeling) is the another simulation environment which works similarly like petri nets, Casual Block Diagram. In this paper we are implementing YAWL in AToMPM together with its operational semantics. We first go through introducing the project and related work. In the next section we define YAWL and its tools, AToMPM and Implementing YAWL with AToMPM operational semantics. Then in the final section will conclude the work and the path for future development.

1 Introduction

Design languages are the main perspective for software developing. Then it will differentiated to many unique modeling languages like python, statecharts, BPM(Business Process Management) , Yawl and ATomPM. These

*

†

modeling languages ensures efficient and effective business operations through software making [1]. By this discipline the developers can be capable of developing a trust worthy software and analyzing tools. Every organization woks under some kind of well disciplined work flow systems. In some cases they do collaborate with other entities to improve the efficiency of their businesses. The simulation techniques perform main role in researches as well for handling a large amount of data and calculations and its not a simple process[2]. On the other hand bring up one simulation language to another is an interesting part.

This paper explores the transformation modeling languages for another level of simulation. Taking the simulation in to another modeling with its semantics can be an newer approach to let it adapt in to it. Generally every design languages has own set of working capacity and proficiency when dealing with business models. Precisely for the organizations an research oriented development it expresses variety of unique results. Analyze the large amount of data and work flow pattern, we need sophisticated modeling mechanism so that the newer modeling languages are very much coordinated with the companies [2].

This paper addresses the possibilities of design pattern in other and initiate the way for the future ideas either. The idea is to make sure the languages can adapt with other, so that it will increase the chances improvising the simulation terminology. The domain specific languages like AToMPM has the space for in-building new tools with its semantical way of dealing the model[13]. These are two different environment doing somehow similar job(paper DSL forge). The furthers steps present the related work, YAWL work flow pattern, AToMPM, implementation of YAWL in AToMPM and concluding with this article.

AToMPM is an online graphical modeling environment, which uses a subset of UML for the definition of modeling language and renders model elements(Lajmi, Martinez & Ziadi, -2014). AToMPM is similar to petri nets and state charts, purely simulation oriented system with the collection of formalisms and models[13]. It is also one of the design pattern which are sectored in to steps. It is working based on the predefined formalisms helps to design your own patterns and tools. Works based on the meta modeling layered structure[13].

2 Related Work

The work has developed similar to our approach is Petri nets are modeled and transformed by AToMPM simulation language[10]. Petri nets is one of the know designing language, which is more base for other modeling environment. Every step in petri nets is sequentially designed building process which makes the transitions work well. Many simulation languages are more similar to petri net model, so it gives an impression to the users that main design languages were developed with the help of petri nets. The beauty thing in formalisms are we build one in another with the help of their tools, to let the languages work on their same style. AToMPM is the simulation formalism has the ability to incorporate with some fewer modeling languages. [13]

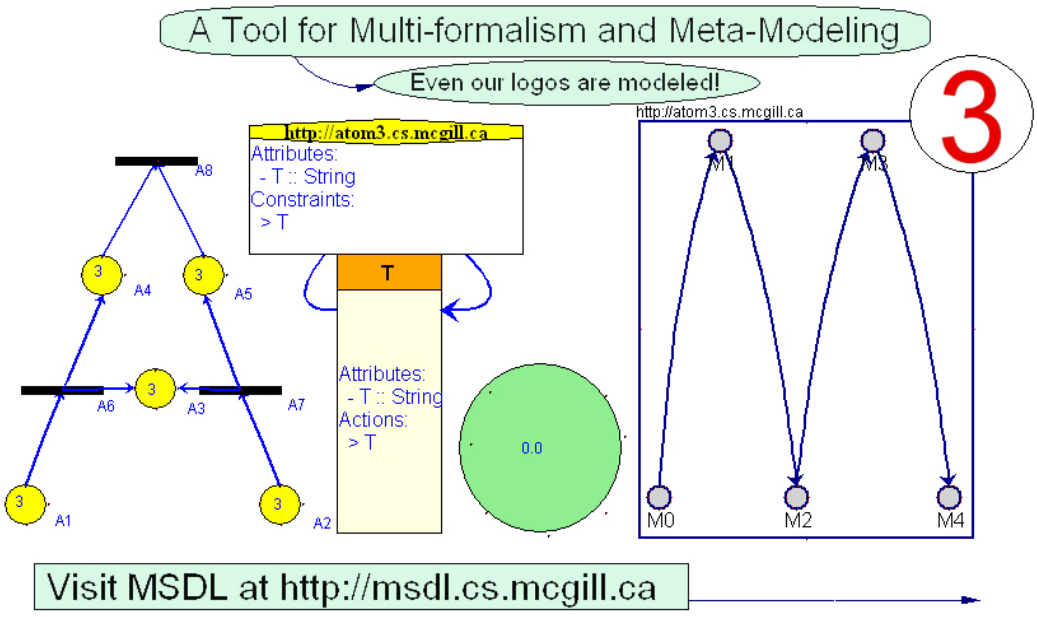

Figure 1: Petri nets in AToMPM

To make the newer simulation languages to be able to model with AToMPM, there are some internal applications which helps to build upcoming techniques. Besides petri nets Casual Block Diagram(CBD) also used in AToMPM for the modeling purposes. CBD is one of the modeling technique used for simulation and constraint oriented methodologies. Such methods are part of the modeling world to be able to cope up with the certain level of abstracting techniques from other mechanisms[12].

For the pet nets formalism the transitions, token, place, Arc and inhibitor are the main tools for making connections and transactions. To develop these tools in AToMPM was an challenging task to do so. The next part is developed by developing metamodel, models and transformation. By the end of this part we are able to use the petri nets in AToMPM formalism. We chose this example to briefly explain the similarity between what we had and what we going to implement. There some more game examples also exist for our references [4][5].

3 YAWL Workflow pattern

YAWL system is one of the most mature open-source workflow management system available at present(understanding user references). Wikipedia saids that YAWL is an extended form of petri nets and improvised design environment[7]. YAWL support for the business process management and organizational utilities which can deal with tons of data and applications for the industries. It has been part of the medical sectors which helps to analyzing, testing the data level abstractions[8]. For computing software in a simple way and easy-to-use approaches the YAWL system is used. It support for the high level graphics, application designs, deployment, execution and monitoring for the software systems. Enabling the business development for the organizations and research oriented sectors as well. The step by step design used to analyze the data, deep overview of the software components, functionalities of entities used. It exhibits the clear view of the related information from the user point of view [2].

It has an comprehensive support for the control-flow systems. YAWL contributes a large impact in modeling world with its operational and application level design patterns. It is a service oriented architecture, so any language can adapt with this system. It supports passing files as data and organize those data by this internal engine. The data's are sequentially scheduled by the YAWL engine. To adopt the YAWL in other language is a lot of work, so that it split in to three different sub parts. They are Control panel, Editor and Engine[6][9].

3.1 Control Panel:

Its a small desktop application supports for starting, stopping, updates and status. It shows the engine's status either engine is running or stopped.

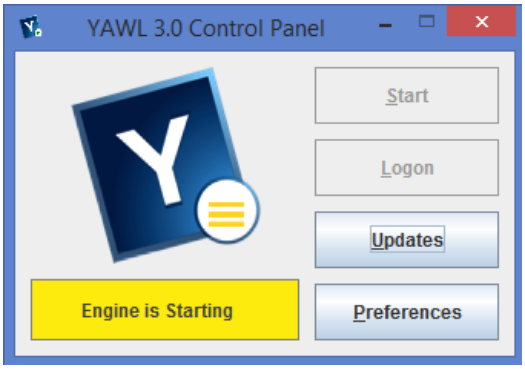

Figure 2: Control Panel

3.2 Editor:

Its design window with the controls and tools. It is used for creating, deleting, configuring, editing, validating and analyzing work flow specifications. The window with tools and settings together.

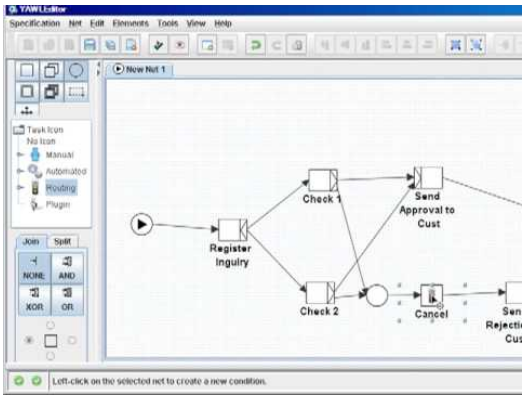

Figure 3: YAWL Editor

3.3 Engine:

The YAWL engine is used for execution, once we done with the design works. This is mostly for manging, manipulating and organizing the models.

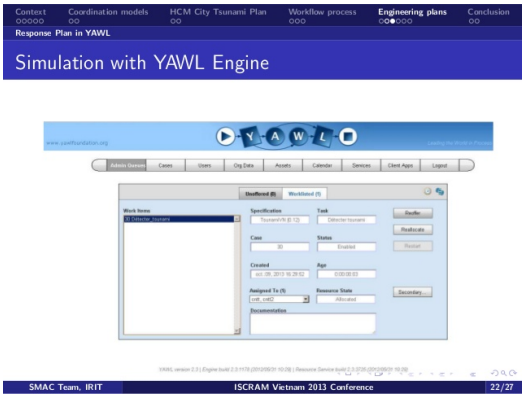

Figure 4: YAWL engine

4 Defining Tools

:

In this section covered by YAWL tools and its use in the design perspectives. YAWL has an unique tools used to model the architecture. Each and every modeling language has their own different tools to handle the design patterns. Every tool in the diagram below has the special use. YAWL tools performs the main operation in YAWL work flow. This helps to designing, Editing, Modulating, make connections and executing [8]. The user can design depending on their organizational requirements and needs. It supports multiple platform to work with.

Condition: The tool condition is used for the defining constraints. It works like if... then..else... way to check the condition is true or not. Depends on the output it reacts.

Atomic Task: This function performs like transitions in petri nets. To place an attribute or a value it supports. It is an action based tool runs an operation.

Composite Task: It defines the set of atomic task are called composite task.

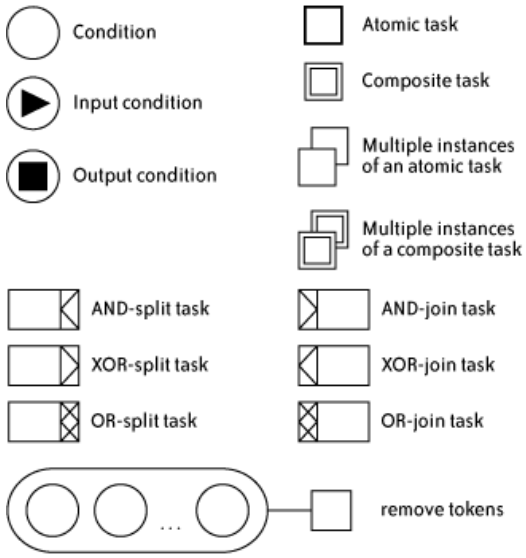

Figure 5: YAWL Tools

Arcs: The tasks and conditions are connected by the arcs.

Input Condition: The input condition to initiate the design. It is a starting point for the model.

Output condition: It helps to stop or end the operation. This tool used to complete design or model.

AND-Split and AND-Join: This function is an one of the atomic task performs an AND operation, the two condition need to be true, then it executes. Split and Join decides to separate the task or join the task.

OR-Split and OR-Join: It performs the OR operation like either...or, one of the condition is true it executes. Depends on the split and join it decides to split or combine the operation.

XOR-Split and XOR-Join: The XOR split is used to trigger only one outgoing flow. XOR join performing an internal merge for the incoming signal. If some conditions if true it executes, but in a inverted way.

Cancellation: This tools helps to remove the tokens from the model. It is used to cancel the activity and case.

In our case of study we are combining the two web based modeling language. So, by performing the transforming actions the whole language need to be changed in to another. But its longer process, so that we decided to start changing with tool in AToMPM. The next section will explain about

AToMPM and how this process will be implemented?[9].

5 AToMPM

AToMPM is a web based modeling technique, which uses UML as a subset for the definition of modeling language. It is one of the visual modeling environment was developed by Eugine Syriani from MCGILL University, Canada. It works under local host over on-line through its client and server. It is a tool for multi paradigm modeling for building metamodels, transformations and for executing Formalism Transformation Graph(FTG)[5]. In AToMPM the languages like petri nets, statecharts and CBD are metamodeled using the internal formalisms. The models are synthesized in AToMPM as abstract and concrete models. Every model works with AToMPM are metamodeled and modeled as abstract syntax and concrete syntax. AToMPM uses the UML as the main formalism to develop the other languages in to it. It uses class diagram, entity diagram, relational diagram to design. It is an improvised version of AToM3 which can be called as a successor of AToM3[4].

AToMPM runs on a web browser and provides the coordination for distributed collaboration in real-time. So the data and design can be accessed only over the web [12]. It support for Domain Specific Language(DSL) to collaborate with the different domain languages. The language is totally depends on the formalisms and simulation work flows. So, the internal operations requires the appropriate terminology to move on to further steps. Every level works under some rules, for example it starts by design class diagrams and transitions between them, transformation to graph model, scheduling the design and generating code. The following figure shows the general view of AToMPM window and some toolbars[10].

Visual modeling contains two different semantics 1)Operational Semantics and 2)Denotational Semantics. The operational semantics is the process of transforming the defined model in to their own look(Symbols for the languages) and performing some simulation experiments. Denotational semantics are the further development of previous progress. It uses the petri net formalism to create transitions and token for the concern language we want to develop. Then it uses the simulation techniques to assign the appropriate action for the symbol we created. In our approach we are going do the similar operation for the YAWL work flow patterns. Further section describes how do we bring up YAWL in AToMPM formalism.

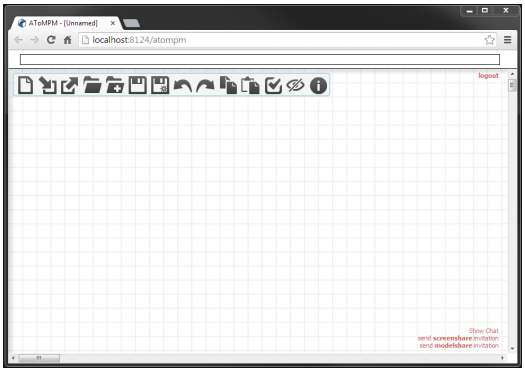

Figure 6: AToMPM editor

6 Transforming YAWL work flow in AToMPM:

The goal of this paper is to implement one modeling language in to another. So, we are transforming YAWL language in AToMPM with its semantics[3]. YAWL is quite a big environment to deal with. And also it has three different sections to run it. Then we decided to design YAWL tools in AToMPM with its operational semantics. The plan is to make YAWL tools works in the same manners as it is in another language. First we start by the design a class diagram for the YAWL tools. It is actually called as abstract syntax, for that we have to use pre defined formalisms to generate the class diagram [11]. The user can create, edit, delete and save the model with the help of some tools. The new design contains all of its tools which shown in the section Defining Tools. There are in-builded tools of AToMPM are lead us to develop the model. This model is called as metamodel of YAWL tools. The following figure shows the actual design. The arc between the class diagrams are transition between two action, for example from atomic task condition, condition task, and so on.

The arc defines the dependency of the another class diagram. There are some constraints like conditons AND-Join has the dependency from condition class diagram. There are some conditions also applied for using the arcs. It has two choices, the options are for being dependent with neighbor class or not. We need to explore the YAWL in AToMPM, so depends on the requirement of tools we can change our choices.

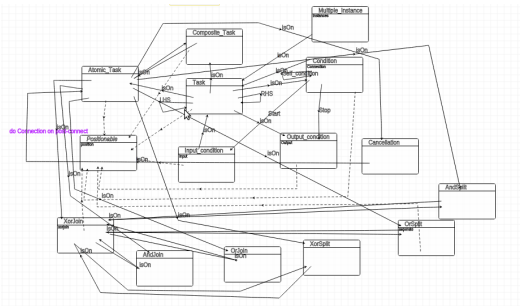

Figure 7: Class Diagram

7 Building concrete syntax:

Next step is to creating concrete syntax for our metamodel. We had to load some tool bars to model them. In this step we are giving a face(symbol) for every tools [13]. The same images are used to define their functionalities of every tool. Tools are having unique actions and performances. Due to that cause we had to use the precise symbols for each tool. The following diagram shows the view of concrete syntax.

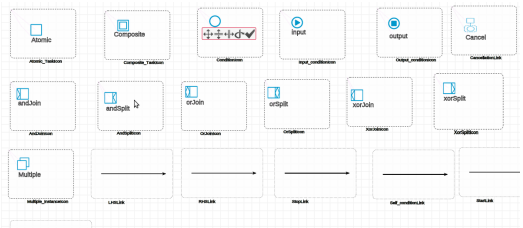

Figure 8: Concrete Syntax

The tool has been placed in that dotted square box. The pop up window appears to ask for the options for the positioning our tool. By this progress for every class supposed to create a symbol to let them work. This model will be compiled by the compiler tool of AToMPM and saved as YAWL.defaultIcons.metamodel. Once this is done to do the transformation, we needed to compile our metamodel to patterned metamodel. While compiling with compilation tool, the window pops up for select the our original abstract metamodel. We need to select out YAWL.metamodel and then click Ok, then it creates two new files in the same folder which helps

to create the transformation. The files are YAWL.pattern.metamodel and YAWL.defaultIcons.pattern.metamodel. The patterned Icons looks with the hash(#) symbol on top left corner of the Icons(tools of YAWL).

8 Transformation:

The next step is to create a transition to each other nodes, so we had to create an action between two nodes to connect. This needs to be done for all the possible connections between YAWL tools. The process start with load a tool bar of trasformation by $FormalismsTransformationTransformation_rule$/Tranformation.defaultIcon The below diagram shows how the transformation tools look like.[10]

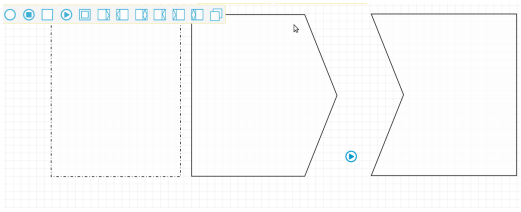

Figure 9: Tranformation of YAWL

The two boxes are LHS(Left Hand Side) and RHS(Right Hand Side) and the dotted box represents the (NAC)Negative application conditions [10]. LHS represents before the transition and RHS represents after the transition between two tools. And the NAC represents the negative action like in the case of there is no transitions happened.

In the next example shows that creation o input condition for YAWL. For this we needed to use th pattern icons to give as input to the transformation tool[11][12].

8.1 Creation of Nodes:

In this part of transformation, the rest of the tools has to be specified and modeled for the purpose of the tool. Every icons or tools must be initialized to make the tool to perform the appropriate action[10][6].

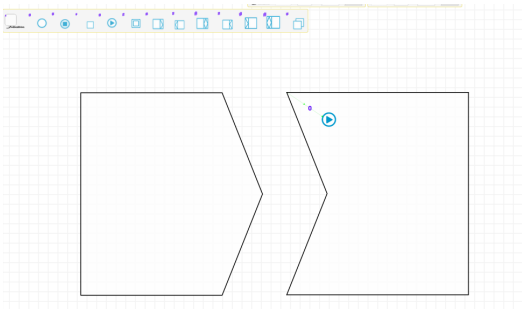

Figure 10: Creating input condition

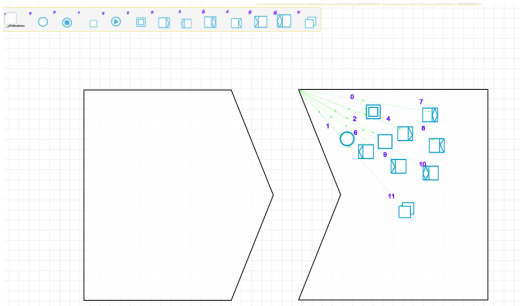

Figure 11: Creating nodes

8.2 Connect Nodes:

In YAWL workflow all nodes are connected visually. So the connection between two nodes since from start node till the end node. If there is no link between nodes then the model will be considered as a wrong model[12]. The connection for all the possible direction were modeled in transformation to perform the action similarly as YAWL workflow. Every actions are modeled in the same way as before by using the transformation tool[12].

There should be one thing is every model should end with the output or end condition. Without the output or end condition the model is not fulfilled or the whole model.

Once the all the actions were created the next step is to play with the creation actions. This way the workflow can be checked consistently. The next section will explain more about the process have been done[10].

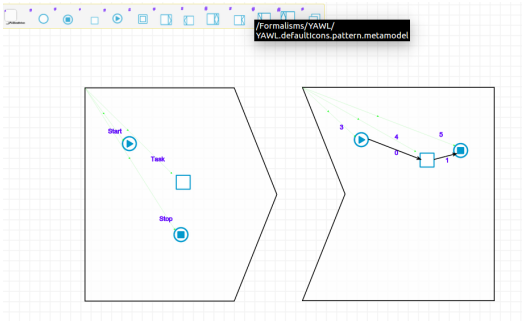

Figure 12: Connecting nodes

9 Simulation:

In this simulation process the simulation must start from where the model begins with. In simulation the black dot tool is the starting point of the simulation to start[13]. So in YAWL the model start with the Start conditions. For the start condition we used Q-Rule. Under the rule green tick symbol confirms the action succeeded; the green cross symbols confirms the action fails. So below the every rule the tick and cross symbol makes sure the model executes further or not. Then we created the B-Rule, is sub-transformation for all the actions. So we place all the actions in the B-Rule. For the actions we created A-Rule for every single transformation action we have created in the previous section[10]. For the end condition we have created the Q-Rule

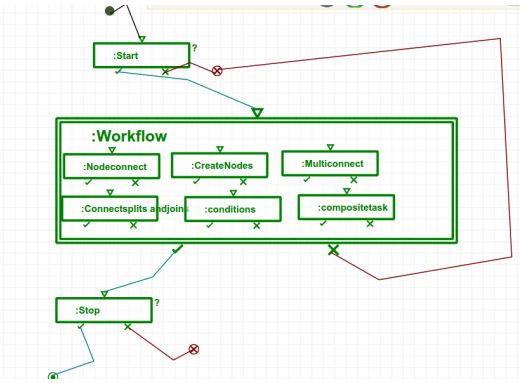

Figure 13: Simulating the workflow pattern

as well to end the simulation. The circled green dot confirms the positive end

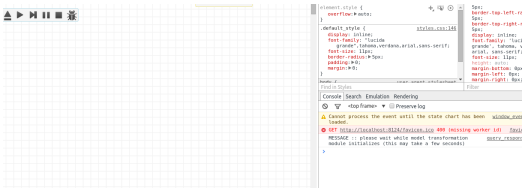

Figure 14: Running transformation

for the model. The red cross symbol executes once the action failed. The next figure shows that the running of simualtion by using the transformation controller button. We need to load our target model to run the transformations. This process is to check the model has been created finely and check our simulation part works well. In this procedures the operational semantics of the YAWL has been developed to tell the modeling world and developers that AToMPM is capable of creating another tool[10].

10 Conclusion and Future work:

In this paper we presented the way to create modeling languages in another. YAWL work flow pattern is an higher level of modeling which dealing with many researches and industrial needs. On the other hand AToMPM is another visual modeling working on the web browsers. The research shows the possibility of creating and merging language together. Even though we experimented that to make the path for the future innovative ideas. The idea is to implement YAWL in to AToMPM with its operational semantics was achieved with this work. There are lot of newer modeling environment need for the business processes. YAWL is a broad and cool model application to perform analysis, business process and modeling, so we considered to develop the tools first, then the further development will be able to design the entire work flow language in AToMPM.

References

[1] Wynn,M.T., De Weerdt,J., Ter Hofstede,A., van der Aalst,W., Reijers,H.A., Adams,M., Ouyang,C., Rosemann,M., & Low,W.Z.(2013).

Cost-Aware Business Process Management: A Research Agenda. Australia, Brisbane:Queensland University

[2] Nguyen, T., Trifan,L & Desideri,J-A.(2010). A Distributed Workflow Pattern for Simulation. France, Grenoble:Project OPALE,INRIA Grenoble Rhone-Alpes & France, Sophia-Antipolis, INRIA Sophia-Antipolis Mediterranee

[3] Lajmi,A., Martinez,J., & Ziadi,T.(2014). DSLFORGE: Textual Modeling on the Web. France, Paris: Universite Pierre et Marie Curie

[4] Mustafiz,s., Denil,J., L'cio,L., & Vangheluwe,H.(2012). The FTG+PM framework for Multi-Paradigm Modeling: An Automotive Case Study. Belgium, Antwerp: University of Antwerp & Karel de Grote., Canada, Montreal: McGill University

[5] Mustafiz,s., Denil,J., L'cio,L., Vangheluwe,H & Jukss,M .(2013). FTG+PM: An Integrated Framework for Investigating Model Transformation Chains. Belgium, Antwerp: University of Antwerp & Karel de Grote., Canada, Montreal: McGill University, School of Computer Science

[6] Recker,J.(Ed.)., & La Rosa,M.(2012). Understanding User differences in open-source work flow management system usage intentions. Australia, Brisbane:Queensland University

[7] Wikipedia: YAWL Workflow

[8] Mans,R.S., & van der Aalst,W.M.P.(2013). Supporting the Workflow Management System Development Process with YAWL. The Netherlands, Eindhoven: Eindhoven University of Technology

[9] Chen, Z.(2012). Workflow Management Theories and Techniques including the Dat Perspective. Netherlands, Twente: University of Twente

[10] AToMPM: http://www-ens.iro.umontreal.ca/ syriani/atompm/atompm.htm

[11] Ergin, H., Syriani, E.(2014). AToMPM solution for IMDB case study. United States of America, Tuscaloosa AL: University of Alabama

[12] Corley,J.(2013). Debugging for Model Transformation. United States of America, Alabama: University of Alabama Tuscaloosa

[13] Jukss,M., Vangheluwe,H., & Verbrugge,C.(2012). Implementing Graph Transformation Languages using RDF storage and SPARQL Queries. Belgium, Antwerp: University of Antwerp