RPG modelling with Web GME

Raha Naseri

Department of Computer science, University of Antwerp

Abstract

In this report I will discuss the modelling of RPGame using a novel modeling tool called webGME. WebGME is a web-Based cloud-based tool which simplifies the design procedure of the system. This tool employs Model Integrated Computing (MIC) approach and is very suitable for online collaboration and large scale projects due to its version control. In this project I tried to get as far as possible in modelling RPGame utilizing webGME tool and outlined my experience and compared it with my previous experience which was modelling RPGame using the AToMPM tool.

Keywords: WebGME, metamodel, domain specific modeling language

1. Introduction

Considering the increase of systems complexity, implementation of these systems becomes harder and harder. Here is where the modeling tools can play a very efficient role in breaking down the complexity. The visualization of these tools are helping to simplify the implementation and analysis of the complex systems as well. WebGME is one of these tools where the user can implement the metamodel and concrete syntax of the model. Different features of it like crosscuts give the user the ability to analyze the project from different point of views. This tool is platform free and can be accessed immediately from the browser without any installation of the tool or dependencies. The rest of the paper is structured as follows: In the first section I describe the plan of my project. Section two outlines some of the advantages and disadvantages of the tool. In section three and four I described the implementation of my project and in section five I discuss the limitations of

Email address: raha.naseri@student.uantwerpen.be (Raha Naseri)

the tool which prevented my project to be completed according to my initial plan. Finally I provide a comparison between webGME and AToMPM in section six and in the last section I provide my conclusions.

2. The Initial Plan

After the first presentation I decided to complete my project according to the schedule shown below. Initially, Since it was not possible to implement all the steps of the project in webGME, I had to design only the metamodel and the concrete syntax of RPGame in webGME, the next step was to implement the rule-based transformation in webGME which required pIcons of my RPG model elemenets and the rule-based transformations modeling language. For this purpose I had to model the rule-based transformation in webGME by creating its metamodel and concrete syntax, and make the pIcons of my RPG model elements. Then utilizing them I had to build the rules of RPGame in webGME. Furthermore I had to create the schedule for the game and execute it, but since this was not possible in webGME I had to export my project from webGME and use another tool or method. Henceforth, since the output file of webGME is a json file the idea was to convert this json file to python and using a compiler similar to AToMPMs compiler to convert these models and rules into Himesis graphs. Himesis graph is the core of AToMPM. The strategy in mapping to himesis graph is that each node(object) of the game is mapped into a himesis node or graph and the association relations are mapped to edges of himesis graph. The himesis nodes have certain attributes which store the attributes of the models elements. This makes it very simple to find the matches in rule-based transformation's LHS. Furthermore these himesis graphs are executed using py-T-core, what py-T-Core does is that it finds matches to the corresponding Himesis graph of LHS in the model and it changes the attributes of objects to their new value according to the corresponding Himesis Graph of RHS. Finally I had to get the result of the execution and print it in the form of text but I could not reach these levels of the project.

. The main obstacle to complete this project according to the initial plan was the fact that I could not have any association relation between an association and an Element in webGME. This happened when I tried to impute a label to every element of my RPGame to imitate the pIcons labeling in AToMPM. My idea was to create an object called label with an integer attribute Num

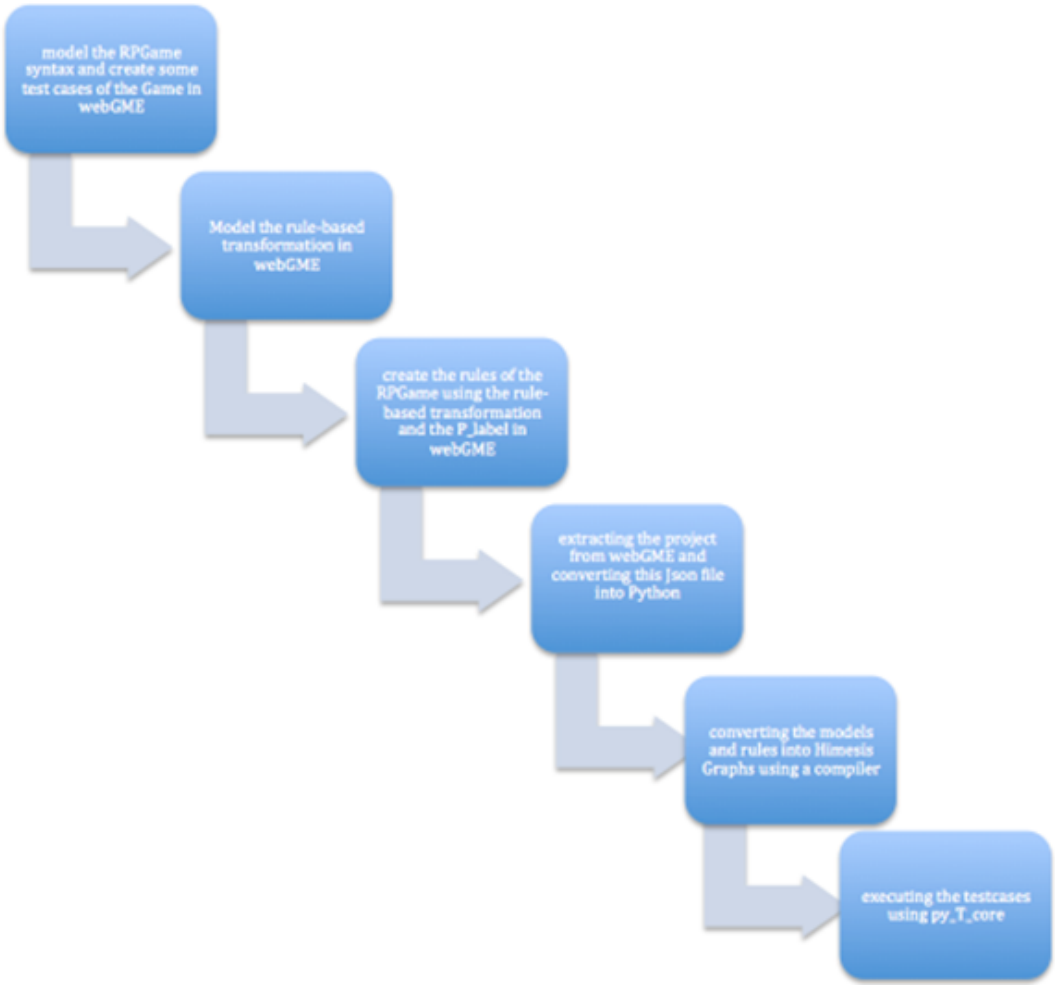

Figure 1: The general plan of modeling the RPGame using WebGME

which represents the number of label and assign one instance of this object to every element in the Rule. This was not possible since it was not possible to create an association between, for instance contains and the label of it.

3. Advantages and Disadvantages

There are some interesting advantages in webGME design which makes it a modern reactive and scalable Domain specific Modeling tool. Here I mention some of them according to my experience with this tool:

Advantages of webGME:

1. Protypical inheritance : Every model element in webGME is a prototype which can be instantiated. An instance is a deep copy of the model, where there is a dependency relation between the original model and the instances. Changes in the original model propagates to the instances immediately.

2. Version control:The version control is done by treating all the model as read-only and updating references whenever a something changes. It only stores the changes and it propagates it to all the users who are using that model concurrently. Nevertheless the user does not have to download the whole database but only the part that he is working on.

3. Collaborative online:Collaborative online working is very well supported, since all the changes made in the project is applied and visualized to all the online users immediately. Using the repository the users can easily fork and branch the project and work in parallel, or analyze any version of the project without interrupting the other users. The project can reach any version of it, this is also very effiecient to return to the previous states when something goes wrong in the project and start over from that point.

4. WebGME is more dynamic than AToMPM from updates aspect. Generally updates and changes in webGME propagates immediately to model and metamodel, but in most of other modeling tools it needs to be compiled and/or saved manually.

5.No seperation line: Using the inheritance the metamodel is not completely separated from the DSML.In other words there is no specific separation line between Meta section and Composition section of the tool. Every change in metamodel propagates the change in the models immediately.

6. Database accessibility:Server gives a scalable access to the models database. However it is not necessary for the user to download the whole database but only a small fraction that he is working on. Hence in this way by occupying the minimum memory space the user can access all the database.

7. Offline working: Offline working is possible in webGME, since the frac-

tion that the user is working on is downloaded, user can make changes offline and these changes can be uploaded whenever the connection is enabled again.

Disadvantages of webGME

There are some issues in webGME that show that this tool is not yet mature enough to be used and that the developers have to make a lot of improvement on it. I point out some of these disadvantages here :

1. To create multiple models with a specific language the user has to model everything in one default composition page. According to the related papers[1] it has to be possible to use every model element as the building block for another language. In order to do so, the user has to double click on the model element in composition page or composition hierarchy and a new page for that element opens. Practically it is not possible to model anything in that page because when it opens all the elements of part browser disappear(even the FCO which is used as the seed to create different elements of the modeling language). Moreover it was not even possible to drag and drop from the composition hierarchy.

2. Although it is written in the related papers[1] that it is possible to import multiple model languages and merge them in the crosscuts of the tool but practically it is not. When the user tries to import a model along with another model the first model disappears.

3. It is not possible to sign in to the tool with a personal account, since there is no way to sign up or make an account. The user has to log in by using a default account called demo.

4. It is not possible to import any icon or image in the tool, to create the icons for the model elements there are only a very limited choice of images in the tool, which can not be resized or edited as well. Instead there are some not very useful and handy options in the main toolbar for changing the color of texts and borders or managing the route of relation arrows and crossing bumps.

5. The inheritance relation is fixed. What I mean is that it is not possible to penetrate in the inheritance tree and add or delete an element in the

middle of this hierarchy. For example at one point when I found out that I need to add a superclass for all the objects and I tried to insert this object in the inheritance tree without recreating its subclasses, it was not possible since the metamodel only displays these already existing inheritance relationships; they cannot be edited so I had to restart designing from scratch.

6. The Meta section in webGME is a powerful environment. To design the abstract syntax of RPGame although it still has some weak points. For example, the attributes of the objects can have limited types, like string, integer, float, boolean.

4. RPGame

My previous experience of modeling RPGame using Modeling tools was with AToMPM where the Game had some entities like Door,Key,Weapon,Obstacle,Scene,etc. But since I found out that webGME is not a very convinient tool for designing RPGame.in figure 2 I have shown an overview of modeling RPGame in AToMPM.

For simplicity I eliminated these parts, so in the basic version of RPGame which I designed in this project there are only the necessary elements like Tile, Hero, Villain, Goal.

5. Implementation

Considering that I have already described the different parts of user interface of webGME in my reading report, I will give a slight extra description here. There are 2 main partitions in this tool for metamodel and the model of the project. Meta is an environment to design the metamodel using the UML classes and relations and Composition is where we can design our concrete syntax and models.

a) Modeling of RPGame

Initially I started opening a new project, this window is where I can open an existing project or import one or create a new one.

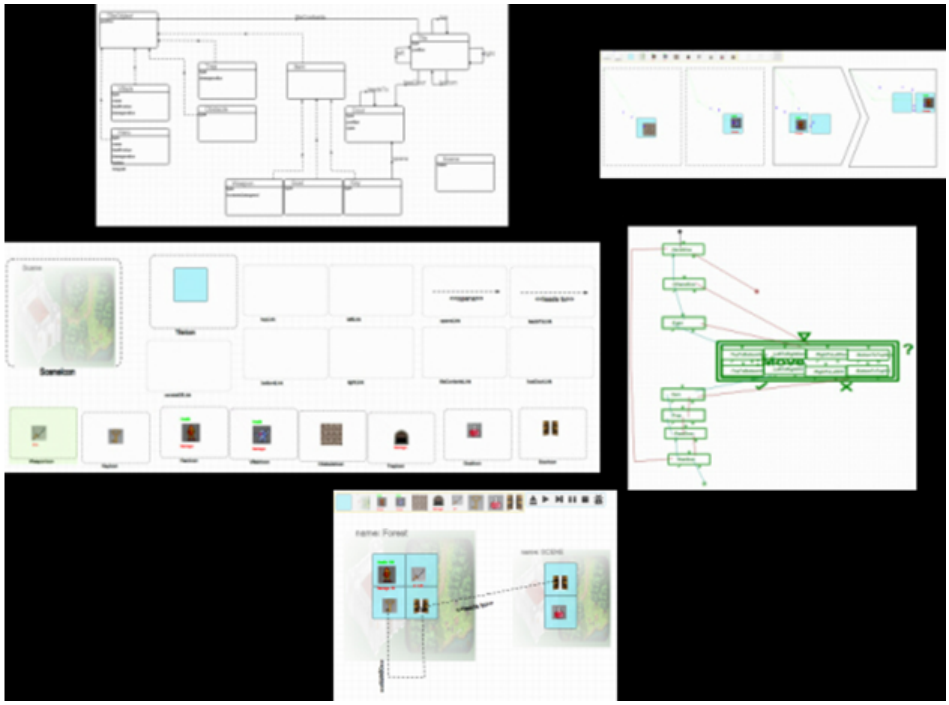

Figure 2: RPGame Modeling procedure in AToMPM

As we can see in figure3 the user here has access to all the projects in database.

The implementation starts with a seed model element called 'FCO'. Due to the prototypical inheritance property of webGME the model's element are all instances of a Root element(FCO). In order to create new classes I dragged and dropped 'FCO' into the canvas and start to change the attributes like name of it using the property editor on the right side of the interface. If I want a B element to be a subclass of A element than I will just drag and drop the A element and change the name attribute of it to B. In that case of course B element will inherit all the attributes of A, so any change during the project in element A in property editor or metamodel will also change it in B element immediately. Now when I created the objects of the project I switch to Meta view to edit the relations, add attributes, constraints, aspects etc of these elements. In order to visualize the elements in the Meta view, I have to drag and drop those elements from the composition hierarchy tree on the right top side of the interface. We can see that the inheritance relation

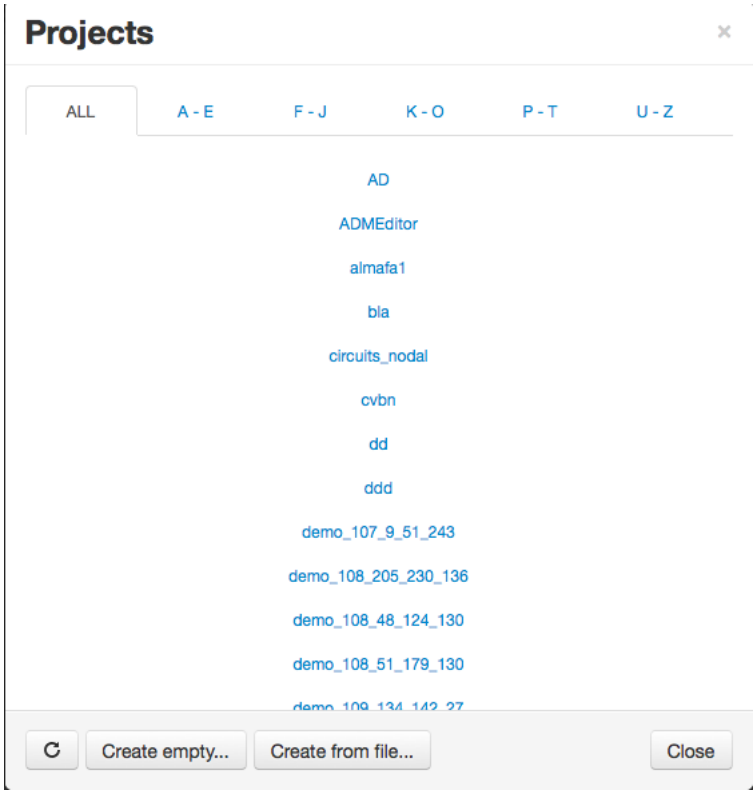

Figure 3: Open new Project

shown with red line is established automatically, and it is not editable. The relations that can be used here are the pointer relation, containment relation and set relation which are respectively represented with blue line, black line and pink line. Association rules are also considered as objects in webGME so to establish an association relation we have to create an object and then set the source and destination of the association with pointers in the Meta view. As an instance a 'Tile Contains TileContent', so to represent the relation 'contains' between 'Tile' and 'TileContent' I made an object called 'Contains' and drew a src pointer from 'Contains' to the source of the association('Tile') and a dst pointer from 'Contains' to the destination of the association('TileContent'). Since 'Contains' is an association it will not be shown in the part browser but it will be displayed as a connection in the model. In the metamodel there is a filter option where the user can filter different kind of relations like pointers, containment, etc, this is very useful

for the analytical purposes.

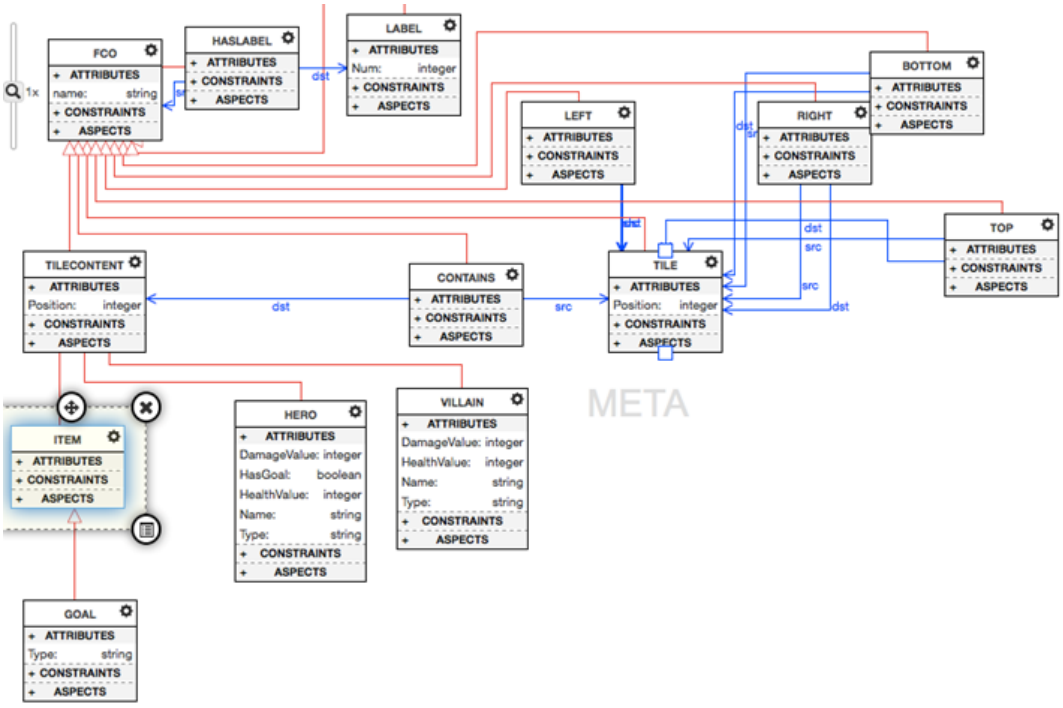

Figure 4: MetaModel of RPGame in WebGME

After completing the Metamodel I return to the composition view and start editing the appearance of the elements, as mentioned before importing icons and images is not possible in webGME. As a result I tried to choose the most similiar images from the limited choice for the icons of RPGame model elements. Figure 4 demonstrates the concrete syntax of the RPG.

After completing the abstract syntax and the concrete syntax of the RPGame, by using the part browser I can design my test models. For clarity I did not stick the tiles of the game so that their names and associations are visible. This is a simple test-case containing all the elements of my RPGame.

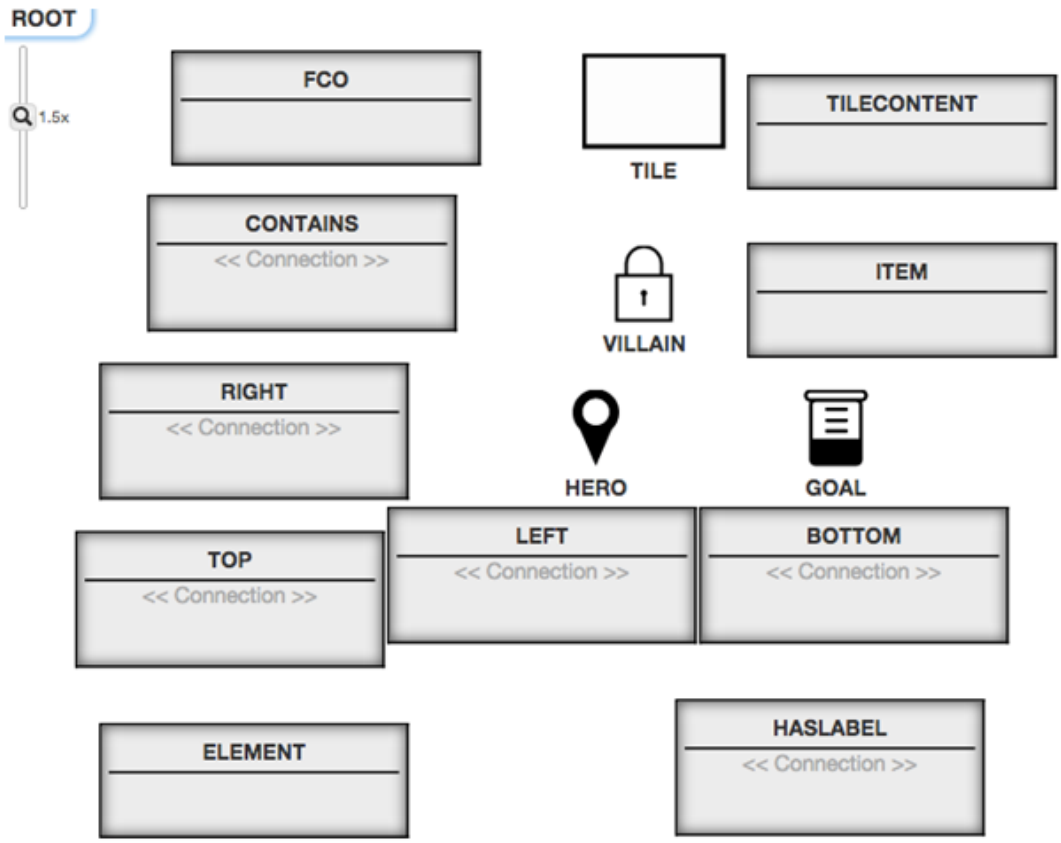

Figure 5: Concrete syntax of RPGame inWebGME

b) Modeling of Rule-based transformation

The next level as I mentioned earlier is to design the Rule-Based Transformation Language,and create the pIcons of RPGame model elements where the elements are associated with a label. These pIcons are used in the Transformation rules. In order to design the Transformation Language initially I had to create a Pattern object. There are 3 kind of Patterns: LHS, RHS and NAC. LHS and NAC have a condition to be satisfied and RHS has an action to be applied. Thus I gave an attribute called Condition to LHS and NAC objects, and an attribute called Action to RHS. To avoid complexity, I designed the metamodel of the Rule-based Transformation Language in a new sheet called Rule-Based Transformation. It is possible to switch between the sheets of metamodels of the project from the tab on top of the canvas.

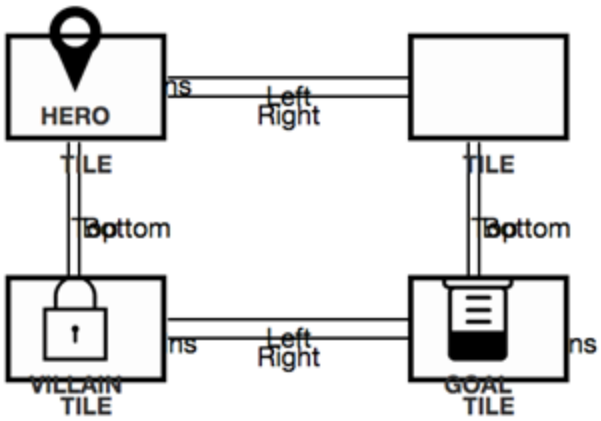

Figure 6: A testcase of RPGame in WebGME

As we can see in this UML, a pattern contains multiple labels so I drew a composition relation between pattern and the label. There is an association between each label and the pattern, I called this attribute Pattern's Label. Now to create labeled icons(pIcons) for RPGame elements I had to make an association between the model elements and the label hence I had to add the object 'Label' to the metamodel of my RPGame and create an association called HasLabel between 'Label' and 'FCO'. In this way all the children of 'FCO' including the model elements and the associations could have a label if necessary.

Furthermore I tried to design the attributes and appearance of the model elements of Rule-Based Transformation Language and again due to the limitations of images, I tried to choose the most appropriate images for each element. The following figure shows the concrete syntax of the rule-based transformation. By setting the display format to 'Num' I could make the label be displayed by the value of its Num attribute.

After completing the concrete syntax of the rule-based transformation language, I had the appropriate infrastructure to design the rules of RPGame. The rule that I implemented was a simple rule, where the Hero wants to move from a tile to another tile on the right side of it. This rule consists of one NAC pattern, a LHS and a RHS pattern, the LHS is the case that

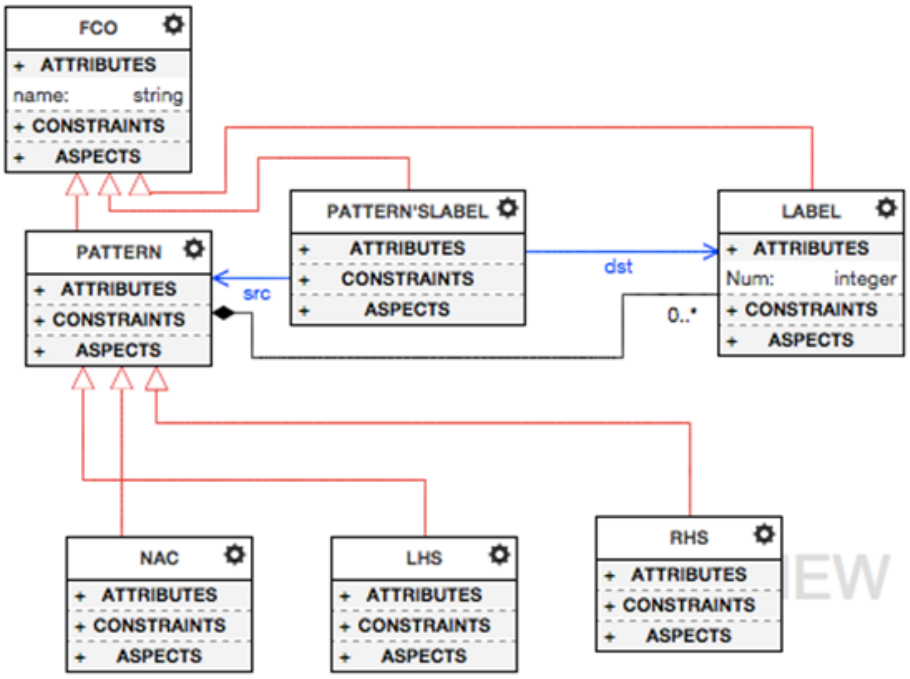

Figure 7: MetaModel of Rule-Based Transformation in WebGME

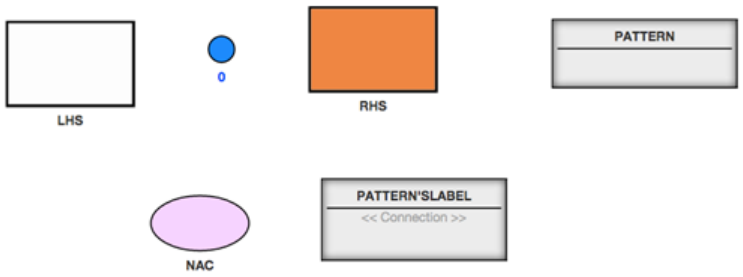

Figure 8: Concrete syntax of Rule-Based Transformation inWebGME

the hero is in the left side tile and the right hand side is the case where the hero is in the right side tile, but this will happen in the case that there is no villain in the right side tile, so as you can see in figure 9 I tried to implement this as the Negative Application Condition(NAC) of the rule.

This stage was actually were I met an impasse and further progress became

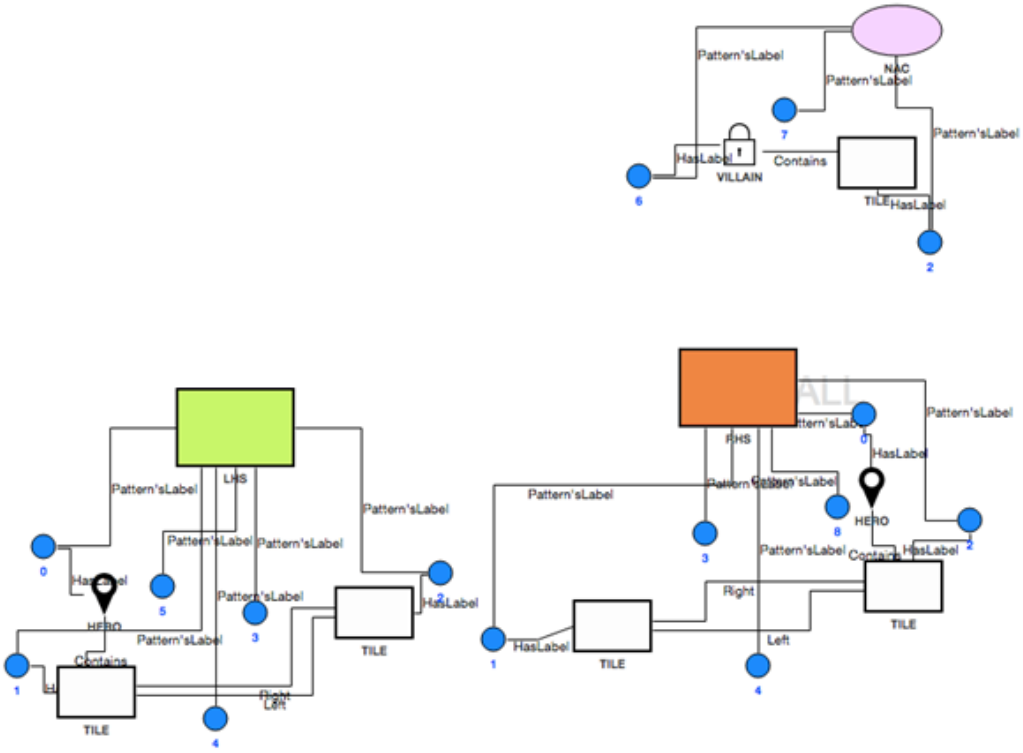

Figure 9: A Rule of RPGame in WebGME

halted for me.

6. The Obstacle

According to a related paper authored by the developers of the tool [1], connections are ordinary models; they can contain children, have other pointers and can be derived, etc. Therefore, the connection concept as such is not part of the meta-metamodel.
According to this, what I tried to do was to assign a label to every existing object of RPGame in the rule including the Model elements and the associations, I had to do this by drawing an association which I designed earlier in the UML called has label between the objects and their corresponding label. This was possible for the entities like Tile or TileContents but it was not possible for the associations. Although according to its UML since the 'HasLabel' association's source is FCO and all the associations are children of FCO,

so it has to be possible to have any kind of the association of the RPGame as the source of HasLabel.
In the following figure you can see the LHS of the rule,where as an example the association Left has to be associated with the corresponding label of it number 4.

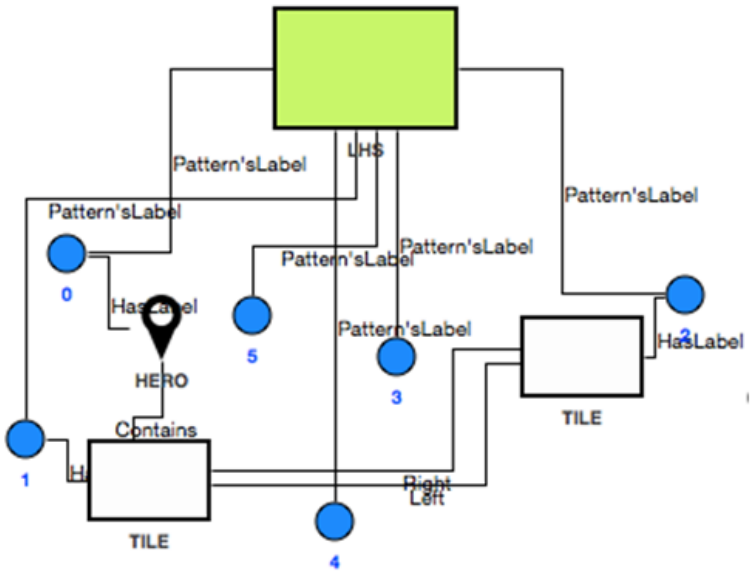

Figure 10: the LHS of a rule of the RPGame in WebGME

7. ATOMPM vs webGME

Personally what I suppose is that in RPGame modelling AToMPM outperforms webGME. Here are some comparisons between these two tools:

- AToMPM environment is a complete package where the user can implement all the stages of modelling including abstract syntax, concrete syntax, rule-based transformation, creating execution schedules and executing the model whereas in webGME environment the user is only able to implement the first two stages.

- While AToMPM is able to support complexity it has a very plain structure, it is much more user friendly and everything is visualized as much as possible.

- It is not possible to import images in the tool or edit the existing limited images in webGME while this is possible in AToMPM.

- In AToMPM to generate the model elements and the metamodel, the UML has to be saved and compiled while in webGME this is done automatically.

- Online collaboration in AToMPM is more static and not very efficient while this is implemented excellently in webGME using the version control.

- To work with AToMPM the user initially has to download the database but in webGME has only to download the small fraction that he is working on at a time while he can access any part of the database.

8. Conclusion

As is plainly evident webGME is not the best choice to model an RPGame. This is mostly because of its immaturity and incompleteness. On the other hand what I believe is that generally webGME is more efficient and useful for large scale projects where many users want to work and analyze it simultaneously. Despite the weaknesses and its lacks, I have to mention that it supports the version control and online collaboration without fail and it has handy options to get different views in different aspects from the project. This is very useful for analyzing the project from different aspects. The filter and the aspect option in metamodel and crosscuts are some of these options.

References

[1] M. Maroti, T. Kecskes, R. Kereskenyi, B. Broll, P. Volgyesi, L. Juracz, T. Levendoszky, A. Ledeczi, *Next Generation MetaModeling: Web- and Cloud-based Collaborative Tool Infrastructure*,Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA.

[2] M. Maroti, T. Kecskes, R. Kereskenyi, P. Volgyesi, A. Ledeczi, *Online Collaborative Environment for Designing Complex Computational Sys-*

tems,Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA.

[3] E. Magyari, A. Bakay, A. Lang, T. Paka, A. Vizhanyo, A. Agarwal, G. Karsai, *UDM: An Infrastructure for Implementing Domain-Specific Modeling Languages*,Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA.

[4] URL: https://github.com/webgme/webgme/blob/master/docs/tutorial.html

[5] URL: https://www.youtube.com/watch?v=0YCo4cpoB7k

[6] URL: https://www.youtube.com/watch?v=7eFP75n5lTY

[7] URL: http://www.isis.vanderbilt.edu/tools/GReAT