MASTER OF ENGINEERING
PROJECT REPORT

EPSILON
ASTAH-GSN
DRIVER

Baran Kaya

400284996

kayab@mcmaster.ca

CONTENT

ABSTRACT

1. INTRODUCTION

Epsilon Astah-GSN driver project is my McMaster University Master of Engineering project. The main goal of the project is to use the Eclipse Epsilon Model-Driven Engineering tool to modify Astah GSN models. The integration between these two programs made with the Epsilon Model Connectivity Layer (EMC). This project is developed for General Motors.

Epsilon is an Eclipse plugin for Model-Driven Engineering processes. Epsilon website defines it as “Epsilon is a family of mature languages for automating common model-based software engineering tasks, such as code generation, model-to-model transformation, and model validation, that work out of the box with EMF (including Xtext and Sirius), UML, Simulink, XML and other types of models.” [1].

GSN stands for Goal Structuring Notation. GSN Working Group defines it as a visualization of safety argument elements and relations. Requirements, claims, evidences, and contexts are some of the safety argument elements [2]. GSN aims to show these elements and the connection/relationship between each element. Astah GSN is one of the modeling programs that specifically designed for creating and modifying GSN models.

Explain sections!!!!

2. RELATED WORK

3. PROJECT REQUIREMENTS

The main goal of the project is to be able to use Epsilon on Astah GSN models so that users can work on the Astah GSN models. In order to do that, the driver to be able to parse the Astah GSN files correctly. Astah GSN saves its model files with '.agml' extension; however, it provides XMI import/export features for GSN models. Astah GSN encodes AGML files so users cannot reach these files content. All EOL functionality can be used within other Epsilon languages as well. Therefore, in this driver project XMI files used. Also, there was Plain XML driver for Epsilon and its used as a based version of the project driver.

Epsilon has one core language (Epsilon Object Language, EOL) and ten task specific languages. Task specific languages derived from EOL thus, integrating models to EOL is sufficient for all other language support. The first requirements show the EOL integration and the later ones show task specific language integrations.

1. Users should be able to load Astah GSN models into the Epsilon in Eclipse IDE.

2. Users can read/access the GSN models with EOL.
   2.1. Users should be able to access the entire GSN model
   2.2. Users should be able to access certain types of elements
      2.2.1. Users should be able to access all nodes
      2.2.2. Users should be able to access all links
      2.2.3. Users should be able to access certain types of nodes (e.g. goals)
      2.2.4. Users should be able to access certain types of links (e.g. asserted evidence)
   2.3. Users should be able to access specific element
      2.3.1. Users should be able to access specific node
      2.3.2. Users should be able to access specific link
   2.4. Users should be able to access elements' attribute values
      2.4.1. Users should be able to access element's content
      2.4.2. Users should be able to access element's ID
      2.4.3. Users should be able to access element's type
      2.4.4. Users should be able to access element's XMI:ID
      2.4.5. Users should be able to access element's XSI:TYPE
      2.4.6. Users should be able to access link's target
      2.4.7. Users should be able to access link's source

3. Users can update the GSN models with EOL.
   3.1. Updating element content

3.2. Updating element ID
3.3. Updating element type
3.4. Updating element XMI:ID
3.5. Updating element XSI:TYPE
3.6. Updating element target link
3.7. Updating element source link

4. Users can create new elements and append these elements into the GSN model.

5. Users can delete elements in the GSN model.

4. DESIGN

As mentioned before, Epsilon Plain-XML driver used as a base for this project. Plain-XML driver can parse XML files and users could load, read and update XML models in Epsilon with this driver. However, this driver is not useful for Astah GSN XMI files. Because, GSN XMI files store every element in the GSN model with the same tag but Plain-XML driver parses files via different tag names. Astah GSN uses XML attributes to store every elements' values. Therefore, Astah GSN driver heavily modified for attribute values instead of tag names.

5. IMPLEMENTATION

In this section implementation of the driver and usage examples are given. Each requirement would be explained with examples.

1. Loading the Astah GSN model into Epsilon

Since this driver based on Plain-XML driver, loading model file code is the same. The only difference between these 2 drivers is names. For instance, "Plain-XML Document" changed to "Astah GSN XMI Document". Other than names rest of the model loading code is work like Plain-XML driver. Below steps explains how to load an Astah GSN model into Epsilon.

a. Create a new EOL file in Eclipse IDE with Epsilon.
b. Right click the EOL file and click *Run As > Run Configuration*.
c. Choose *EOL Program* and create new Run Configuration.
d. Choose your EOL files in the *Source* tab.
e. Go to *Model* tab and click *Add* button
f. Choose *Astah GSN XMI Document* and click *OK* (Figure 1).

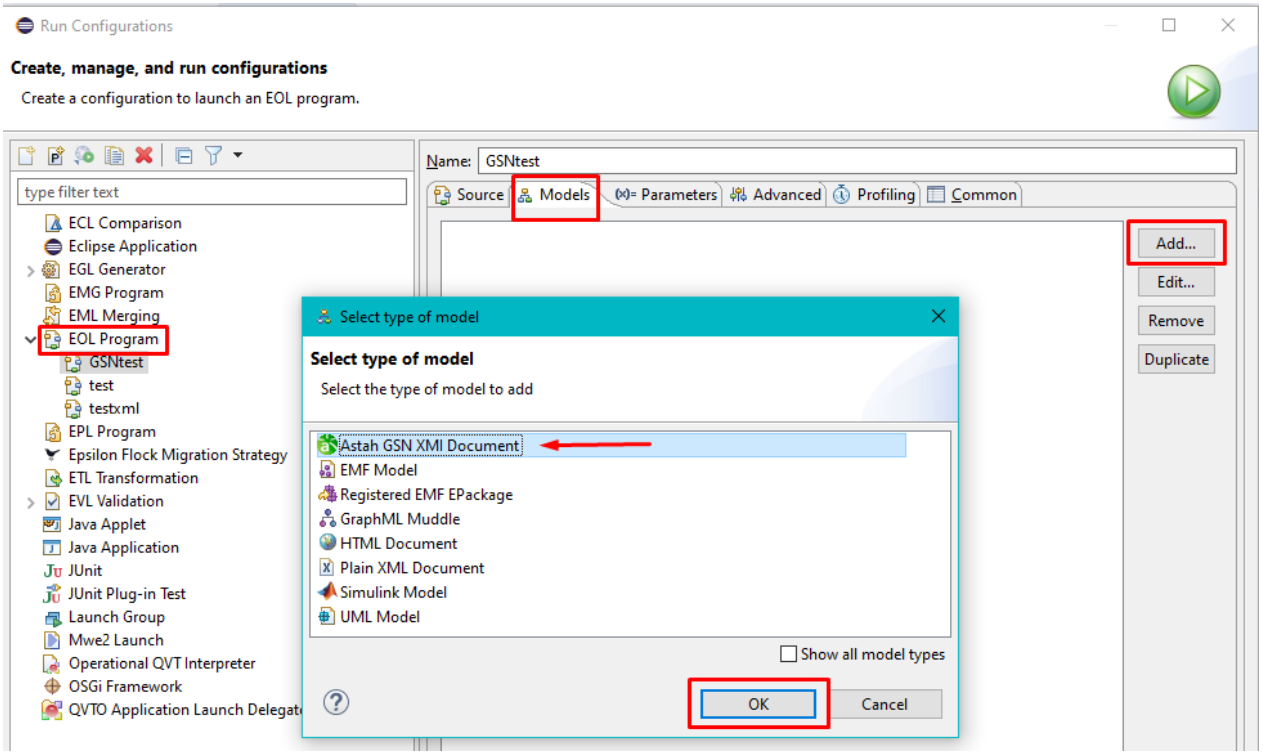

Figure 1: Loading Astah GSN model into EOL

g. Give a name to your model, it is not very important if you don't want to use multiple models in the same EOL script.
h. Choose your XMI file if it's already in the workspace. If not, add your model file into workspace.
i. If you are going to update/modify the GSN model, choose both *Read on load* and *Store on disposal* options (Figure 2).

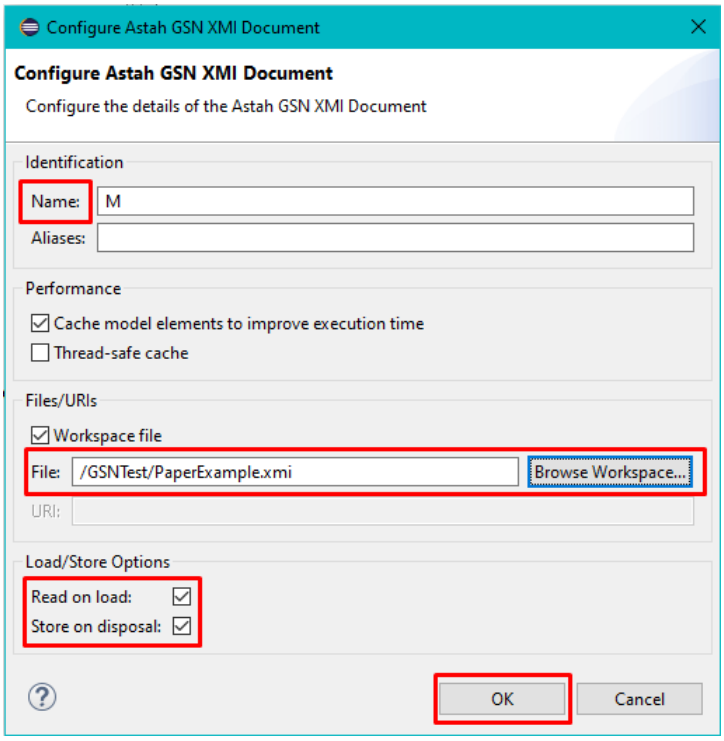

Figure 2: Loading model configurations

j. After that, you can run EOL script with *Run* button. EOL will run on your Astah GSN model (Figure 3).

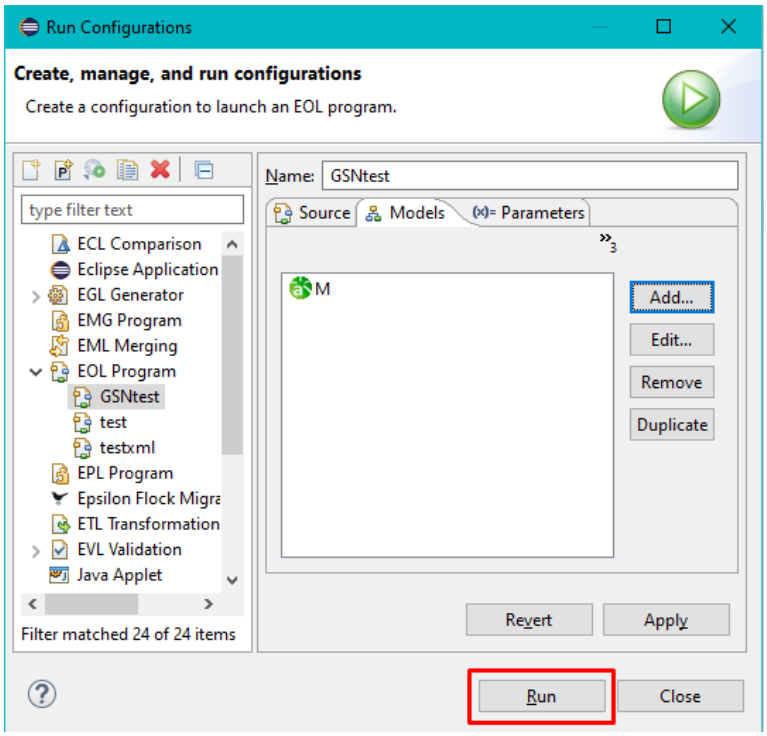

Figure 3: Running EOL with Astah GSN model

2. Reading/accessing GSN models with EOL

Plain-XML driver has 2 classes for getters and setters. Reading and accessing model calls getter class functions. Astah GSN driver's getter class completely changed and it has minimal similarities with Plain-XML getter class.

Most of the changes made for getting attribute values rather than tag names. Plain-XML getter class has method for getting tags, child tags, attribute values, etc. In Plain-XML driver users can only select different tag named elements but Astah GSN XMI document requires getting different attribute valued elements.

Plain-XML documents can have several layered elements so driver can get the root element or any child element. Astah GSN driver have two different options: getting the root element or root's child elements. Because GSN XMI document only has the root element (tag name: ARM:Argumentation) and its child elements (tag name: argumentElement). Some of the methods get root element and some of them only get children. For instance, *gsn.all* call parses the document and returns root element. On the other hand, *gsn.goal* or *gsn.S4* calls parse the document, find the specified child elements and return them as a list or a single element.

Node Elements	Link Elements
• Goal • Strategy • Solution • Context • Assumption • Justification	• Inference (Asserted Inference) • Evidence (Asserted Evidence) • Context (Asserted Context)

Table 1: GSN Element Types

Getter Commands	Command Explanation
`gsn.all`	Returns entire model and all elements
`gsn.nodes`	Returns all node elements
`gsn.links`	Returns all link elements
`gsn.goal`	Returns all goal elements
`gsn.assertedcontext`	Returns all asserted context (link) elements

`gsn.all.content`
`gsn.context.content`
`gsn.Sn13.content`
	Returns specified element/s' content attribute value/s. The results could be Sequence or string depending of an element. Link elements' don't have content attribute so it returns empty string ("").

`gsn.G1.gsntype`	Returns given element/s' GSN type
`gsn.strategy.id` `gsn.c23.id`	Returns given element/s' ID. Link elements' don't have ID thus it returns empty string.
`gsn.S5.xmiid` *OR* `gsn.a9.xmi_id`	Returns given element/s' xmi:id attribute. Each element has unique xmi:id values.
`gsn.j5.xsitype` *OR* `gsn.goal.xsi_type`	Returns given element/s' xsi:type attribute. Same elements (e.g. goal and assumption) have the same xsi:type values.

`gsn.Sn3.target` `gsn.context.target`	Returns link elements that have target value as given node element/s.
`gsn.g2.source` `gsn.strategy.source`	Returns link elements that have source value as given node element/s.
`gsn.t_g1_s_g2`	Returns link element that has target value is G1 and source value is G2.
`gsn.all.last`	Returns last element of given elements list
`gsn.solution.first`	Returns first element of given elements list
`Gsn.S3.content.println()`	Prints given value/s

Table 2: Astah GSN Driver Element Getters

Some of the getters could be combined differently. For example, *`gsn.goal.last.content.println()`* and *`gsn.goal.content.last.println()`* prints the same result. The difference between these 2 commands is simple. The first command gets all goal elements list, then finds the last goal element and prints its content attribute value. The second command gets goal elements list, then gets all goal elements contents and creates new list later it prints the last content in that list. Thus, the first command is faster than the second one because, it doesn't get all goal elements' content attribute, it just gets one goal element's content attribute and prints it.

3. Updating GSN models with EOL

Updating elements commands call setter class functions. Most of the element attributes can be set via below commands. Getter and setter commands are the same. The only difference is setters requires "=" character and new value after equals character.

Setter Commands	Type	Command Explanation
`Gsn.sn13.content = "New content";`	String	Updates given element's content value
`Gsn.g3.id = "G6";`	String	Updates given element's ID (ID attributes must be unique!)
`Gsn.j15.xmiid = "fvLpH5q4Eeqyz11T9RpXrQ";`	String	Updates given element's xmi:id attribute. However, Astah GSN generates unique xmi:id values based on model and element location. Therefore, using this command can corrupt the model file.
`Gsn.c7.xsitype = "ARM:ArgumentReasoning";`	String	Updates given element's xsi:type attribute. Changing xsi:type without changing id might corrupt model file.
`Gsn.t_s1_s_g1.target = gsn.sn7;`	Node element	Updates given link element's target attribute to new node element's xmi:id. New value must be node element.
`Gsn.t_J1_s_G13.source = gsn.J2;`	Node element	Updates given link element's source attribute to new node element's xmi:id. New value must be node element.

`Gsn.a12.gsntype = "goal";`	String	Updates given element's type attribute. Changing xsi:type without changing id might corrupt model file.

Table 3: Astah GSN Driver Element Setters

4. Creating new elements in GSN model with EOL

Creating new element command uses *new* keyword. Using *new* keyword only creates new element object but it doesn't append this new object into the model.

Creator Command	Command Explanation
`var newElement = new goal;`	*New* keyword creates new element with given type.
`gsn.all.append = newElement;`	*Append* command attaches given element into the model file. New element would be the last element in the model file.

Table 4: Astah GSN Driver Element Creator Commands

New element's attributes could be set via two different ways. Either updating the *newElement* object like *`newElement.content = "test";`* or accessing the last element and updating its attributes such as *`gsn.all.last.content = "test";`*.

5. Deleting elements in GSN model with EOL

Deleting an element in EOL uses *delete* keyword. One or multiple elements could be deleted via *delete* command.

Delete Command	Command Explanation
`Delete gsn.G10;`	Deletes given element/s.

Table 5: Astah GSN Driver Element Delete Commands

6. Epsilon Validation Language usage
7. Epsilon Code Generation Language usage
8. Epsilon Transformation Language usage

6. TESTING/EVALUATION

7. CONCLUSION

8. REFERENCES

[1]. UPDATE HERE!!

[2]. Epsilon website, "Epsilon Home", https://www.eclipse.org/epsilon/

[3]. Goal Structuring Notation, “What is the Goal Structuring Notation?”, http://www.goalstructuringnotation.info/archives/category/in-a-nutshell