BWCPS-Generator

Intoduction to the bachelor thesis result of

Nada Chatti

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Ralf Reussner
Second reviewer:	Prof. Anne Koziolek
Advisor:	Dipl.-Inform. Emre Taspolatoglu
Second advisor:	Dipl.-Inform. Jörg Henß

11. July 2019 – 11. November 2019

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

1 Introduction

This would be an introduction to the implementation result of the BWCPS-Generator project. As part of this project, some security concerns of cyber physical systems were considered and integrated here and in the BWCPS-Metamodel Project. An introduction to this integration is found in the git-repository of the project. The BWCPS-Generator is here perceived as an Eclipse-plugin that takes as an input a BWCPS-Model and generates as output necessary projects/classes that best implement the entities in the model. As this project was originally a continuation of the Sensidl project it is still tightly related to it and the structure of both projects is mainly similar as well as the implementation of the user interface.

1.1 Functions

The main functions of the generator are listed below.

[F1] User Interface: take the user's generate command and parameters: path, Language, resource(model) to use.

[F2] Map entity to Osgi/Java-item: map the entities created in the model to the Osgi/Java-item that best implements it according to the following table.

Entity	Osgi	Java
SteamRepository	Deploying Package	-
Stream	Component Context	-
Node Container	Bundle	Project
Node	Component and Service	Class that implements Node interface
NodeType	-	Class and attribute in Node class
NodeLink	Reference to other Services	Security Manager attribute and security related methods in Node class

Table 1.1: Map entities to Osgi/Java-item

[F3] Generate: generate the entities created and add them to the user's workspace.

1.2 Installation

Prerequisites: You have to have Eclipse Modeling, JDK 8+ and git installed. The way to install these isn't going to be further explained. To be able to use the plugin you have to first install SensIDL and Metamodel.

1.2.1 SensIDL

Clone the SensIDL repository in your workspace:
`https://github.com/SENSIDL-PROJECT/SensIDL`

1.2.2 Metamodel

Clone the BWCPS-Metamodel repository in your workspace:
`https://github.com/SENSIDL-PROJECT/BWCPS-Metamodel`

Then follow these steps:

- Checkout to BWCPS-GENERATOR_SecurityMeasures
- Navigate to de.fzi.bwcps.stream.metamodel/model/, open metamodel.genmodel (make sure you are on the generator tab), right-click on the Metamodel genmodel (first row) and click on Generate All

1.2.3 Generator

Finally clone the Generator repository in your workspace:
`https://github.com/SENSIDL-PROJECT/BWCPS-Generator`

Then do following steps to make it work.

- Create an Eclipse Application Run configuration that contains all above mentioned repositories (make sure to click on add required Plug-ins) then run it.
- Create a new Project and a bwcps-conform metamodel. You can find an example in the BWCPS-Generator project in resource/model example/ make sure to copy all three files in your project.
- Right-click on the My1.entity file and choose BWCPS>Wizard...
- In the wizard choose Kura Project as language then click Generate.
- Wait for the magic to happen.

1.3 Implementation

The imlementation of the functions mentioned above is as follows.

1.3.1 User Interface

The plugin is first invoked through its menu entry in Eclipse's Package-Explorer with text "BW-CPS". Two commands are here available "Generate Now..." which uses the last settings for generation and "Wizard..." which opens a wizard that takes the user's command. Using these commands would lead the wizard to call the generation handler that would load the resource needed and call the Generator's doGenerate method that leads us to the next function.

1.3.2 Map entity to Osgi/Java-item & Generate

These two functions are done simultaneously as follows. The StreamRepositoryManager class takes the resource as an input and delivers a StreamRepository instance that contains all data that is in the model after validating it against the metamodel constraints. According to this instance the generation steps would be decided.

First the Node Configuartion Project would be generated with or without the security package as needed. Then for each node container following generation steps are added:

1. Project generation step: where the project(that represents an Osgi-Bundle) and its structure is created with necessary metadata.

2. DTO generation step: where java classes and sensidl files are created and saved in the list of files to be generated.

3. File generation step: where that list is generated and if it is a SensIDL file, SensIDL would then be invoked to create the Data Classes that a Nodetype needs as its input and output.

These steps are initiated in the GenerationJobFactory and started in GenerationJob.

1.4 Questions / possible further work

During this thesis a few questions were raised and a lot of solutions/workarounds were found. Here are some questions/suggestions that were still not answered. -Should the metamodel have more constraints. Example:
Nodelink:

- source Node1

- target Node2

Node1.nodetype :

- output Datatype1

Node2.nodetype :

- input Datatype2

It is safe to infer that Datatype1 = Datatype2.
Isn't this a logical constraint that must be in the metamodel?
Constraints like this would make it a lot easier to have an even more accurate generator.