

# Eclipse Scheduling and Network Delay Analysis Plugin

Gustavo Hidalgo

July 6, 2015

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Eclipse Modeling</b>	<b>1</b>
<b>3</b>	<b>Meta Model</b>	<b>2</b>
<b>4</b>	<b>Implementation</b>	<b>2</b>
4.1	Main Project	3
4.2	Edit	4
4.3	Editor	4
4.4	Sirius Editor	4
4.5	Eclipse Extensions	4
<b>5</b>	<b>Example</b>	<b>4</b>
<b>6</b>	<b>Installation</b>	<b>4</b>
6.1	From Source	4
6.2	From JARs	4
<b>7</b>	<b>Future Work</b>	<b>4</b>

## 1 Introduction

This report describes the design and implementation of an Eclipse plug-in for the purpose of designing real-time systems based on the ARINC 653 standard for integrated modular avionics and ARINC 664 for network connections. The plug-in defines a meta-model for a system which can be used by several algorithms to analytically ensure schedulability stability and an upper bound on the network delay for frames sent over a network. The plug-in also has a comprehensive validation suite which prevents users from creating structurally invalid configurations along with a graphical editor for the network definition which greatly helps to describe and debug network configurations.

This document contains hyperlinks to online resources. View it on a computer to follow the links for more information.

## 2 Eclipse Modeling

The Eclipse Modeling Framework is a tool for designing and manipulating models in Eclipse with many facilities for code generation and persistent data manipulation. It is important to be familiar with this framework in order to understand this project. The most important features used are:

1. **Ecore tools** : A graphical meta-model editor
2. **OCLinEcore** : A modified version of the Object Constraint Language used to define structural constraints for an instance of a meta-model

3. **Code Generation** : A large part of the code in the plug-in was generated by Ecore.

4. **Sirius** : A framework for creating graphical DSLs based on Ecore models

These tools are Eclipse plug-ins which must be installed first before modifying the plugin.

### 3 Meta Model

The classes in the Ecore meta-model are described in this section along with the OCLinEcore validation steps that are applied.

For reference, the full meta-model is displayed first and the individual components after.

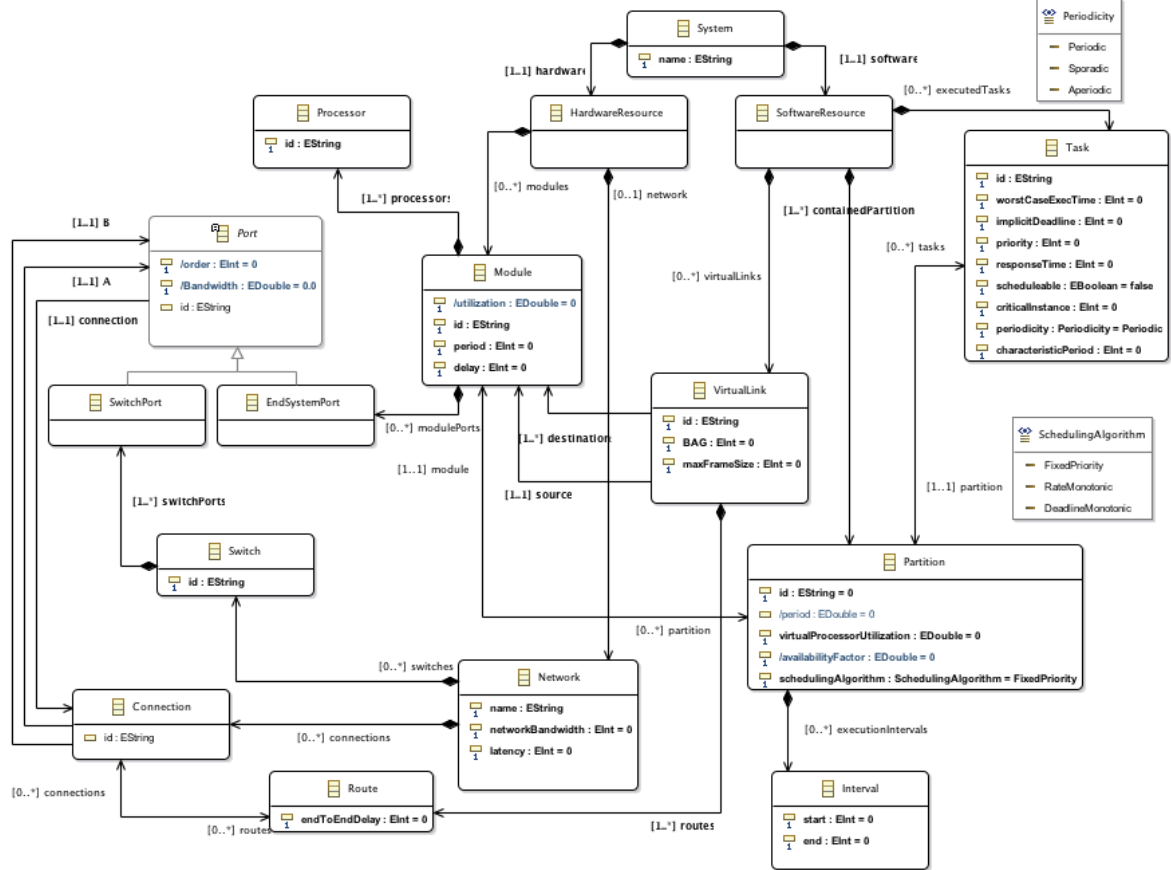


Figure 1: Full UML meta-model

#### System

The root object for an instance of the meta model.

#### Constraints

### 4 Implementation

This section describes Java code and Eclipse specific implementation details of the plug in. The plug in is composed of 5 Eclipse projects, 4 generated from the Ecore model and 1 from the Sirius specification. See figure 2 for the list of projects.

Most of the non-generated code resides in `fr.ensma.realtimescheduling`, specifically the analysis algorithms, visualization classes, and the meta-model interfaces and implementations. This project will be referred to as the **main** project.

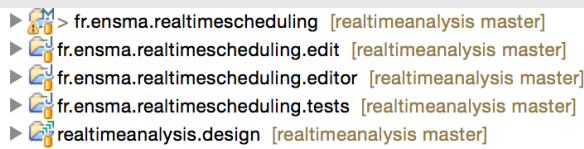


Figure 2: Plug in projects

The projects `edit`, `editor`, and `test` contain *generated* code for editing the model (operations such as instance creation, or attribute modification), the editor GUI (see figure 4), and test code respectively. The test project is currently not used.

Several Eclipse extensions are also declared in the `plugin.xml` within the main project (see figure ??). These mainly consist of UI extensions to allow for easier manipulation and analysis of a model.

## 4.1 Main Project

The main project contains the analysis algorithms and non-trivial UI logic code for the project. The package structure of this project is given in figure 3. Several of these packages deserve special attention.

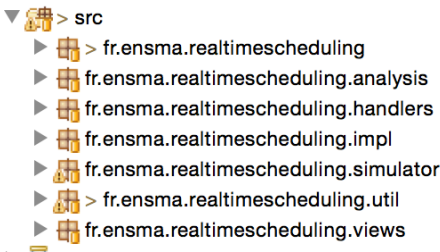


Figure 3: Main project packages

**analysis** This package contains the algorithms and "glue" code between the model and the algorithms.

The `Analyzer.java` class holds the implementations of all the algorithms described in the CORAC-PANDA project: the response time analysis from chapter 2 and the network delay analysis from chapter 3. The code was written attempting to recreate as much as possible the structure and names of the pseudo-code from the report.

The `ModelInterface.java` class contains methods to interface between the UI and the analysis algorithms.

The rest of the classes in the package provide side-effect free functions for manipulating the model or classes to store auxiliary data necessary by the algorithms. For example, the Ecore list implementation `EList` is not modifiable thus it cannot be passed to Java Collections methods such as `Collections.sort()`. Having a list of time-sorted `Intervals` is necessary for the algorithms thus the `PartitionUtils.java` file provides such a method by creating a copy of the original list.

**handlers** This package contains handler definitions for the commands generated by Eclipse. UI actions in Eclipse are handled by a command-handler relationship. More information on this mechanic can be found online.

**views** This package contains the classes used to layout and display data to charts from the model. This project uses the **JFreeChart** library to generate charts like line charts and bar charts. The classes `AbstractLineChart.java` and `AbstractBarChart.java` set up the layout of the charts which is a horizontal layout with a chart on the left and a control component on the right.

**util** This package is generated by Ecore, however it houses the `RealtimeschedulingValidator.java` class which contains the complex validation code that could not be expressed in OCLinEcore. The specifics of the validation are in section 3.

## 4.2 Edit

This project contains generated code for modifying elements of a model instance. The only modifications to this code are found in the `getText()` method of some of the instances which makes the display in the editor more user friendly. See figure 4.

## 4.3 Editor

## 4.4 Sirius Editor

## 4.5 Eclipse Extensions

The plug in provides Eclipse extensions.

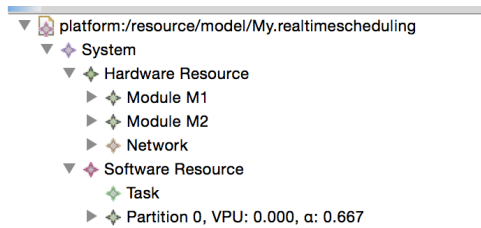


Figure 4: Model editor in use

# 5 Example

This section shows an example model created using the plug-in. I describe how to create it and how to analyze it.

## 6 Installation

### 6.1 From Source

Move the 5 projects into your Eclipse workspace. You can then highlight the 5 projects, right-click on one of them, select **Run As -> Eclipse Application**.

### 6.2 From JARs

## 7 Future Work

This section describes what future work needs to be done to improve the project and the steps required to start that work.