Customizing Papyrus-RT to Facilitate Software Modelling of Rovers

Sudharshan Gopikrishnan
School of Computing
Queens University, Canada
16sg1@queensu.ca

ABSTRACT

This paper is a project report on, how research was conducted to customize an open-sourced software modelling tool called "Papyrus-RT", to facilitate software modelling targeting the Rover case study.

The project provides a customizable configuration page in the multi-editor panel of Papyrus-RT. The configuration page allows one to edit the design configurations of the software model, destined for specific type of rover hardware. The page serve as a single point for all the design specifications and prevents the need for travelling through the model for design changes.

Keywords

Rover Case Study; Papyrus-RT; Open Source; Model Driven Engineering; Real-Time; UML-RT; Eclipse Modelling Framework (EMF)

1. INTRODUCTION

Model driven engineering or (MDE) in short is a software development methodology that focuses on creating and exploiting domain models, which are conceptual models of all the topics related to a specific problem. MDE highlights abstract representation of the knowledge and activities that govern a particular application domain, rather than the computing concepts.

The MDE approach is meant to increase productivity by maximizing compatibility between systems and simplifying the process design. Today, MDE is used in both industry and academia alike [1]. In industry it is used mainly because, business requirements often change faster when compared to the supporting software system. However, MDE can provide a solution as it makes software development faster and, it also easier to change applications (due to it being cost-effective and less sensitive to changes in personnel).

An interesting point worth noting is that, the usage of open source tool and platform in the context of Model driven engineering has become critical. The Model driven engineering tools developed by stand alone companies cannot satisfy the diverse needs and sometimes conflicting expectation of both the industry and academia [1]. Furthermore, proprietary software do not allow for much of customization, and dependency on such software can limit the innovation in companies.

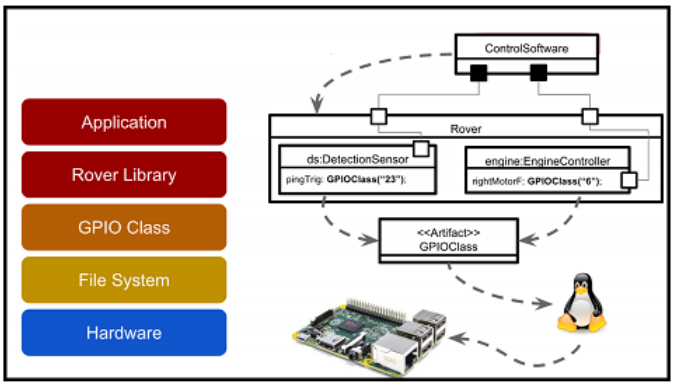

Figure 1: Rover Architecture

Papyrus is an industry grade modelling tool for UML. In recent times, it has drawn attention of both industry and academia due to being open source, highly customizable and the support for Domain specific modelling language. They aim at facilitating collaboration and knowledge transfer between industry and academia. Papyrus for real time (Papyrus-RT)[2] is another open source modelling environment, which is used to model complex real time system using UML-RT language. It comes with an editor for modelling and a code generator for C++. A detailed explanation highlighting the various features can be found here [2].

The project customizes Papyrus-RT by providing a configuration page, which in-turn act as a hub for all the design configurations when modelling for the rover case study. A brief description of the Rover case study is explained in the following sections.

The rest of the paper is structured as follows: Section 2 gives an overview of the Rover Case Study; Section 3 details the software modelling using the Rover; Section 4 highlights the implemented project and finally the last section concludes with the future work .

2. THE ROVER CASE STUDY

The Rover is a small vehicle powered by dual motors; one to move forward and the other to move backward. The rover is designed with minimum functionalities - move forward and backward, and finally rotate. It embeds a set of sensors to detect any obstacles and to collect data from the environ-

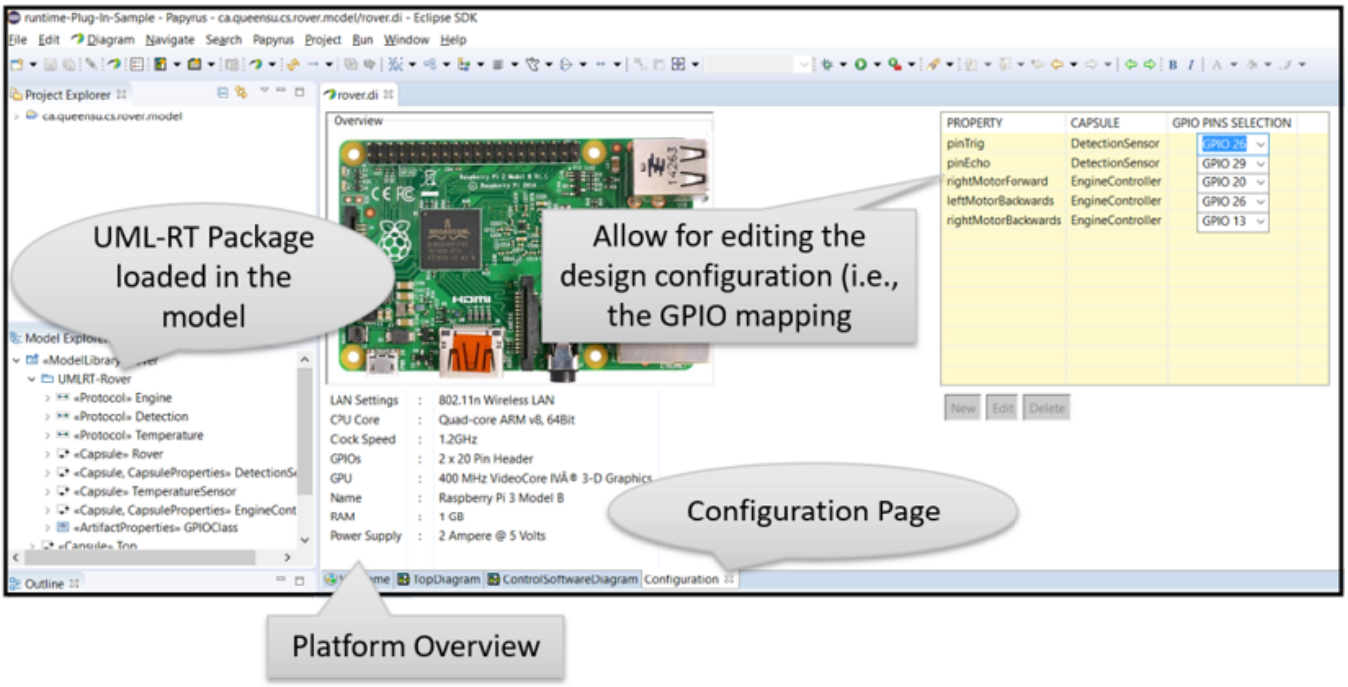

Figure 2: Configuration Page

ment(temperature and humidity). The main processing element in a Rover is a Raspberry Pie platform, which runs on a real time version of Linux operating system. The Raspberry-Pie platform hosts and executes the generated code from the UML-RT model.

The behaviour of the rover system is as follows. In the initial state, it moves forward until an obstacle is detected. To avoid any obstacle, it turns 90 degrees, and then starts moving forward. During the entire span of execution, safety critical information such as the temperature and the humidity is collected from the environment [3].

3. MODELLING THE ROVER

The behaviour of the Rover is modelled in UML-RT using Papyrus-RT. The UML-RT model is composed of interconnected capsules, ports and protocol to model the entire structure of the system. A state machine is encapsulated in each capsule to model its behaviour.

The whole architecture of Rover system consists of five layers of abstraction(see Fig. 1). From the bottom the hardware layer corresponds to the Raspberry Pie, which embeds 26 GPIO pins. GPIO pins stands for general purpose input-output pins. The user interfaces with these pins to get the Rover moving. Among the 26 pin-sets, 17 pins are used to connect to external devices such as sensors and actuators. The file system layer is powered by a real time version of Linux. Each pin in the Raspberry Pie corresponds to a file in the file system. The user interacts with the pins, either by reading values from the file or writing into it. The GPIO layer is a C++ wrapper class for controlling the GPIO pins.

The GPIO class consists of methods to set and get the value of pins and to get the direction of GPIO pins. On top of this, the remaining layers corresponds to the UML-RT modelled using Papyrus-RT. The Rover library layer consists of UML-RT capsule which correspond to different components of the physical Rover. The uppermost layer, the application layer consists of the business logic of the application. Both these layer are connected through Relay ports and a Rover capsule to ensure encapsulation [3].

4. PROJECT IMPLEMENTATION

The goal of this project was to provide a customizable page or a tab (similar to the Papyrus-Welcome Page) in the multi-editor panel of Papyrus-RT. This page provides an overview of the platform and all the design configurations of the Rover under study. For example, in the current prototype the configuration page, (see Figure. 2) would allow for editing the design configurations such as the GPIO Pin mapping discussed earlier. In addition, the user can also alter the attribute names from within the page and need not iterate through the model.

The entire project is a plug-in developed using the developers version of Papyrus-RT. The configuration page (seen in Figure. 2) is the run-time instance of the plug-in project. When the Rover-Package or the UML-RT model is imported to the run-time instance, the plug-in traverses through the model and projects the various configurations of the model, which can be customized.

The Eclipse Modelling Framework (EMF) [4] is used to impart the changes made on the configuration page onto the imported model. This prototype is a single point for all the design specifications and prevents the need for travelling the various state machine diagrams and capsules when we want to change any property associated with it.

5. CONCLUSION AND FUTURE WORK

In the current implementation of the project the papyrus diagram representing the configuration page had been created, with the purpose of serving as a hub for all the design configurations. As a next step into the research, new tabs similar to the configuration page will be created or disposed depending on the different packages (that may represent various features of the platform) being loaded or unloaded.

For instance, within the loaded Rover Package we may want to import additional packages to further customize the platform model, example: packages containing capsules to interface with the Raspberry Pie camera or say a package to add a feature "follow the line", etc. Each of these packages (including the top package "Rover") may contain capsules (aka components) to add some features to the Rover, and therefore may need to be configured (for example, one may be interested in configuring the pin mapping for the Rover platform, another would be interested in configuring the refreshment rate of the camera for a "follow the line" scenario). Design ideas such as the one discussed above are endless. Further research would involve analysing various Raspberry-Pie platforms and coming up with a feature model for them.

Furthermore, the generic mechanism of extension point of Eclipse [5] will be used to register Java classes responsible for creating (and disposing) each tab content for configuring specific part of the platform, depending on whether a specific package (that can be identified by its name or UML annotation or that can be nested in other packages) has been loaded.

6. REFERENCES

[1] Bordeleau,F. and Fiallos,E., "2014 Model-Based Engineering: A new Era based on Papyrus and Open sourced Tooling", In OSS4MDE@MoDELS (pp 2-8).

[2] Papyrus-RT website: "https://eclipse.org/papyrus-rt/"

[3] Ahmadi, Rezza, et al. 'Run-time Monitoring of a Rover: MDE Research with Open Source Software and Low Cost Hardware.", (2016).

[4] Eclipse Modelling Framework: "https://www.eclipse//-.org/modeling/emf/"

[5] Eclipse Extension Points: "http://www.vogella.com -/tutorials/EclipseExtensionPoint/article.html"