Modelling and Code Generation for Real-Time Embedded Systems with UML-RT and Papyrus-RT

Conference Paper · May 2017

DOI: 10.1109/ICSE-C.2017.168

3 authors, including:

Nicolas Hili
Queen's University
19 PUBLICATIONS **51** CITATIONS

SEE PROFILE

Modelling and Code Generation for Real-Time Embedded Systems with UML-RT and Papyrus-RT

Nicolas Hili and Juergen Dingel
School of Computing
Queen's University
Kingston, Ontario, Canada
Email: {hili, dingel}@cs.queensu.ca

Alain Beaulieu
Electrical and Computer Engineering Department
Royal Military College
Kingston, Ontario, Canada
Email: alain.beaulieu@rmc.ca

***Abstract*—This paper discusses the Model-Driven Engineering (MDE) of real-time embedded (RTE) systems with soft real-time constraints using UML for Real-Time (UML-RT) and Papyrus-RT. UML-RT is a profile of UML specifically designed for RTE systems. It has a long, successful track record of application and tool support via, e.g., IBM Rational RoseRT, IBM RSA-RTE, and now Papyrus-RT. Papyrus-RT is an Eclipse-based, open-source modelling development environment for UML-RT systems. It allows the generation of complete, executable code from models and advances the state-of-art via support for model representation with mixed graphical/textual notations and an extensible code generator.**

This paper introduces the central modelling concepts of UML-RT and features of Papyrus-RT. It also presents some advanced features of Papyrus-RT such as import capabilities, mixed graphical/textual modelling, and incremental code generation.

I. Introduction & Motivation

MDE has been proposed as an approach that can tackle the challenges of developing robust RTE systems. Modelling languages such as UML-RT [1] aim at raising the level of abstraction to, e.g., reduce the effort required to deal with lower level concerns such as data serialization, low-level concurrency management, platform-specific aspects, etc. In spite of its proven benefits, the adoption of MDE for RTE systems has been slow [2]. Inspired by the general trend towards open-source software, the lack of open-source, industrial-strength MDE tools has recently been identified as a central impediment to the promotion, adoption, and evolution of MDE [3]. Indeed, MDE tools in the RTE system domain have traditionally been proprietary. Tool users depend on the tool vendor for support, updates, and customization. This dependency has hindered more wide-spread dissemination, not only among industrial practitioners, but also among researchers and educators since it has prevented all three user groups from getting what they need in cost-effective ways.

To promote the development of industrial-strength open source tools for the development of embedded systems, the PolarSys Eclipse Working Group was recently created [4]. The group currently consists of 23 members from industry and academia and supports 24 projects that focus on different aspects of embedded systems development including modelling and tracing. Papyrus-RT [5], [6] is a project within PolarSys; currently in version 0.8, version 1.0 is scheduled to be released in July 2017.

This paper accompanies a Technical Briefing given at ICSE 2017. It aims at giving the readers a very brief view of MDE for RTE with UML-RT and the most recent version of Papyrus-RT. It covers the following objectives: (1) introduce to the core modelling concepts of MDE in UML-RT such as capsules, ports, protocols, and hierarchical state machines; (2) give an overview of the current version of Papyrus-RT; and (3) provide pointers to more detailed descriptions about UML-RT and Papyrus-RT.

II. UML for Real-Time

UML-RT [1], [7] is a profile for UML specifically designed for modelling real-time systems with soft real-time constraints. It has its roots in the definition of Real-time Object-Oriented Modeling (ROOM) [8], a specific language for the development of real-time systems, initially adopted by ObjecTime [6].

UML-RT features a rather small set of concepts. The main concept is the *capsule*, an active class which owns a *state machine* and can exchange *messages* through its *ports*. Ports are typed with *protocols*, a formal definition of the incoming and outgoing messages a capsule can send or receive. Ports can be connected through *connectors* if their ports are typed with the same protocol. In addition, UML-RT only provides two diagrams: *capsule diagrams* representing how capsules are instantiated and inter-connected, and *state machine diagrams* for modelling the behaviour of each capsule. An example of a capsule diagram is given in area 1 of Fig. 1. It shows a top capsule consisting of eight capsule instances inter-connected through connectors.

In addition to the core modelling concepts, UML-RT also features more advanced constructs, such as Service Access Point (SAP) and Service Provision Point (SPP) ports to dynamically bind service providers to clients, capsule and state machine inheritance, optional capsules which are dynamically created and wired at run-time.

III. Papyrus-RT

Papyrus-RT is based on the Papyrus/Eclipse platform and was designed to be extensible, allowing users to add, with relative ease, their own customizations or extensions. Its target audiences are industrial developers who want to build custom solutions, researchers who want to prototype and evaluate

This is a preprint version. See ACM/IEEE for the final official version.
To appear in the ACM/IEEE 39th International Conference on Software Engineering, May 20-28, 2017, Buenos Aires, Argentina.
http://dx.doi.org/10.1109/ICSE-C.2017.168

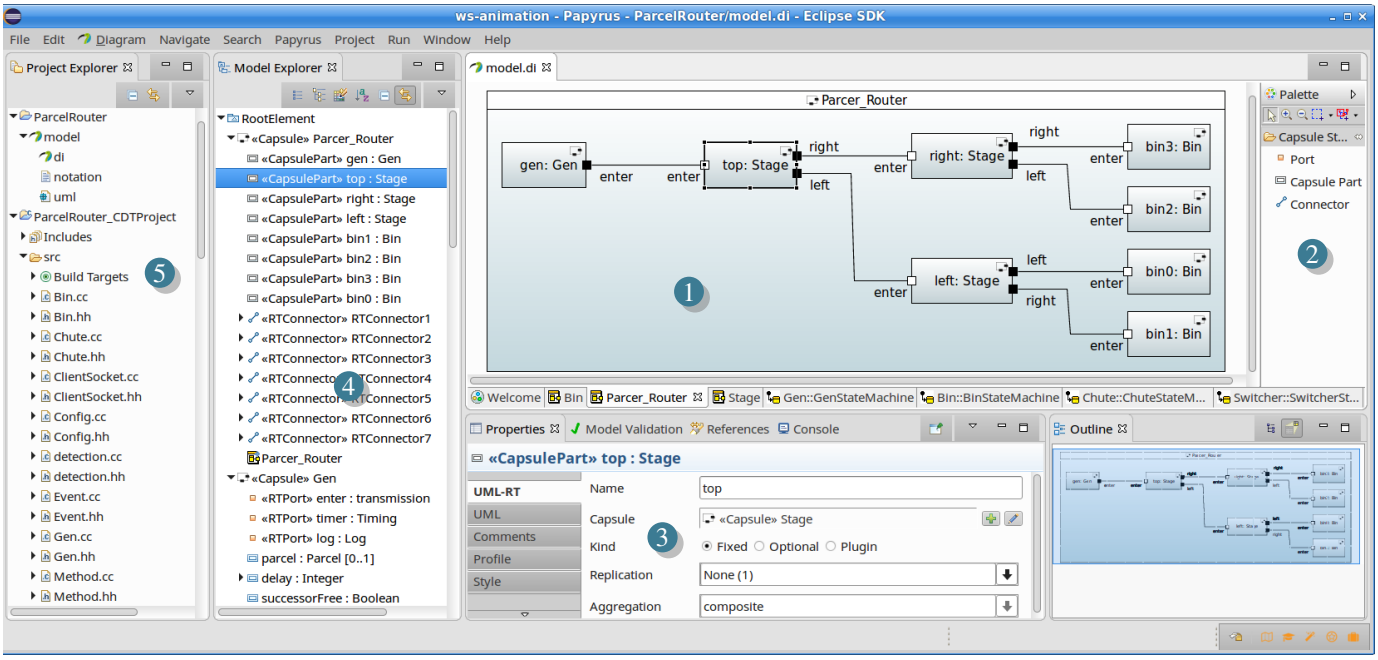

Fig. 1: Papyrus-RT Workbench

novel techniques, and educators who want to teach students the strengths and weaknesses of modelling and MDE.

Fig. 1 gives an overview of Papyrus-RT. The central area 1 is used for graphically editing models through several diagrams. Switching between diagrams can be done using the tab bar below the graphical editing area. A *palette* 2 is used for creating and for adding model elements directly to the diagrams. It has been stripped down compared to the UML2 palette and only shows the relevant UML-RT concepts. Below the editing area, a *property view* 3 can be used for editing properties of the selected model elements. On the left side, a *model explorer* 4 is fully synchronized with the graphical editing area and displays a tree view of the currently edited model. It supports most of the basic actions such as element creation or deletion. Using the model explorer, code can be generated partially or entirely. Projects and generated code can be visualized in the *project explorer* 5.

Compared to previous UML-RT tools, Papyrus-RT provides fully synchronized graphical and textual editors for UML-RT, allowing developers to choose their preferred representation without having to commit [7]. Users can move back and forth between the two representations and code generation is possible from both. Moreover, Papyrus-RT provides full code generation and a run-time system to deploy fully functioning applications on a target platform. Its code generator is incremental, so the entire code does not have to be regenerated every time a change is made to the model. Instead, the code generator isolates the part of the code affected by a change to the model in order to re-generate it. In addition, the code generator was designed to be customizable, for example to support code instrumentation or alternative code patterns. Additional features include code-assist and import capabilities of legacy models from proprietary tools.

More detailed information about UML-RT and Papyrus-RT can be found at [1], [5], [9].

Acknowledgments

This work is supported by Ericsson Canada, EfficiOS, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] B. Selic, "Using UML for modeling complex real-time systems," in *Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES'98)*, 1998, pp. 250–260.

[2] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE Software*, vol. 31, no. 3, 2014.

[3] F. Bordeleau and E. Fiallos, "Model-Based Engineering: A New Era Based on Papyrus and Open Source Tooling," in *OSS4MDE@ MoDELS*, 2014, pp. 2–8.

[4] "PolarSys Working Group Homepage," https://www.polarsys.org/.

[5] "Papyrus for real time (Papyrus-RT)," https://www.eclipse.org/papyrus-rt, accessed: 2016-03-10.

[6] E. Posse, "PapyrusRT: modelling and code generation," in *Workshop on Open Source for Model Driven Engineering (OSS4MDE'15)*, 2015.

[7] E. Posse and N. Hili, "Hybrid Graphical/Textual Modelling and Code Generation with PapyrusRT," in *Tutorial Session at Model-Driven Engineering Languages and Systems (MODELS'16)*, 2016.

[8] B. Selic, G. Gullekson, and P. T. Ward, *Real-time object-oriented modeling*. John Wiley & Sons New York, 1994, vol. 2.

[9] E. Posse and J. Dingel, "An Executable Formal Semantics for UML-RT," *Software & Systems Modeling*, vol. 15, no. 1, pp. 179–217, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10270-014-0399-z