Save-IDE

An Integrated Development Environment for Building Predictable Component-Based Embedded Systems

Séverine Sentilles, Paul Pettersson, Ivica Crnkovic
Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
{severine.sentilles, paul.pettersson,
ivica.crnkovic}@mdh.se

John Håkansson
Uppsala University,
Department of Information Technology
Uppsala, Sweden
johnh@it.uu.se

Abstract— **In this paper we present an Integrated Development Environment Save-IDE, a toolset that embraces several tools: a tool for designing component-based systems and components, modeling and predicting certain run-time properties, such as timing properties, and transforming the components to real-time execution elements. Save-IDE is specialized for the domain of dependable embedded systems, which in addition to standard design tools requires tool support for analysis and verification of particular properties of such systems.**

I. INTRODUCTION

Component-Based Software Engineering (CBSE) is an approach which aims to increase the efficiency in software development by *(i)* reusing already existing solutions encapsulated in well defined entities (components) and *(ii)* building systems by efficient composition (which includes constructive, i.e. functional composition, or "wiring", and composition of extra-functional properties).

To achieve this goal, composition theories are necessary but not enough; the existence of technologies which include tool support is crucial. This is even more true in specific domains, such as dependable embedded systems. The development process for this category of systems requires a strong emphasis on analysis, verification, and validation in order to ensure the necessary quality of the final products. Three types of requirements are characteristic for these systems: *dependability* (reliability, availability, safety, etc.), *timing requirements* (release and response time, execution time, deadline, etc.), and *resource utilization* (memory, CPU, message channels). Further, compared to desktop and pure software systems, software in embedded systems have significantly different design models from execution models; while for example design models can utilize component models, execution models are based on run-time entities such as threads, tasks, scheduling mechanisms, etc. This implies that a CBSE approach for embedded systems must have an additional element in the development chain – transformation of models in the deployment phase. Consequently, there is a need for an Integrated Development Environments (IDE), gathering all the tools and techniques needed in the development process and integrating them with component-based development. Compared to the majority of existing IDEs which focus mainly on the programming aspect, an IDE for component-based dependable embedded systems should highlight *(i)* design, *(ii)* analysis, *(iii)* transformations, and *(iv)* verification.

In this paper we present an IDE developed for the SaveCCM component model [1] as part of a component-based development approach that includes three key activities in the development process: design, analysis, and synthesis. The IDE, called Save-IDE[1], comprises these quite different aspects of the development of component-based embedded systems, and demonstrates an approach to the integration of different tools into a common environment.

II. SAVE-IDE

Save-IDE is designed as a platform and a set of tools that cover the design lifecycle. It includes design of a system and components, specification of component behavior, analysis of timing properties and deployment (simple transformation from components to an execution model, generation of glue code and compilation). Save-IDE is developed in the Eclipse development platform using the Eclipse Modeling Framework (EMF), the Graphical Modeling Framework (GMF), and the Acceleo plugins.

The System Design is realized in an *Architecture Editor*, which is implemented as a graphical user interface used to design SaveCCM systems. The inner parts of compound elements (assembly, composite, and switch) are realized through diagram partitioning that allows distinguishing external view of an element from its internal view. The external view describes the ports, the various models used, whereas the internal view handles the inner components, and their connections. For a primitive component (which does not include architectural SaveCCM elements), only the external view is possible. The internal view is described in the *Component Development* part of Save-IDE.

This work was partially supported by the Swedish Foundation for Strategic Research via the strategic research centre Progress.

[1] The SaveIDE is available for download from the webpage http://sourceforge.net/projects/save-ide/

The Component Development provides both a *Component C-code Editor* and an *Architecture Editor*. The component C-code Editor is used to develop SaveCCM primitive components and is provided by the Eclipse C/C++ Development Tooling (CDT). From the System Design perspective, skeletons for the C-code and the corresponding header files are generated. These skeleton files are then filled with the appropriate source code. The Architecture Editor is also used to develop composite components, which are components constructed out of inner-components.

The exchange of components between the System Design perspective and the Component Development perspective are realized through a *Repository Browser*. Within the System Design perspective, the repository browser also allows the reuse of previously realized SaveCCM elements such as components, assemblies, composites, or switches. Those elements can be located on local or remote repositories.

The Analysis part in the Save-IDE is provided by the *Timed-Automata Editor* (TAE) and the model-checking tool UPPAAL PORT. The TAE provides developers with a graphical user interface for formally modeling the internal behavior of SaveCCM elements as extended timed automata [2]. Using a semi-automatic mapping process it is possible to associate external ports of a SaveCCM element with variables of the internal TA, which allows for formal descriptions of elements composed into composite components or architectural descriptions.

The output of the TAE and the mapping can be compiled (by Save-IDE) to an XML-format accepted by the UPPAAL PORT tool [2] which consists of a graphical simulator integrated in Save-IDE and a verifier. Using the integrated simulator it is possible to validate the dynamic behavior of complete SaveCCM design in the early development phases of a project, prior to implementation. Using the verifier interface, it is possible to establish by model-checking whether a SaveCCM model satisfies requirements specified in a subset of the logic Timed CTL. The UPPAAL PORT verifier is developed based on the timed automata model-checker UPPAAL [3], but extended with partial order reduction techniques which exploits the structure and semantics of SaveCCM model to improve the model-checking efficiency [4]. Fig. 1 shows two screenshots of Save-IDE.

REFERENCES

[1] Mikael Åkerholm et al., "The SAVE approach to component-based development of vehicular systems", Journal of Systems and Software, vol 80, nr 5, Elsevier, May, 2007

[2] R. Alur and D. L. Dill. "A theory of timed automata". Theoretical Computer Science, 126(2): 183–235, 1994

[3] Larsen, K. G., Pettersson, P., and Yi, W. "UPPAAL in a Nutshell". Int. Journal on Software Tools for Technology Transfer 1 (1–2), 134–152, Oct. 1997.

[4] J. Håkansson and P. Pettersson. "Partial order reduction for verification of real-time components". In Proc. of the 5th International Conference on Formal Modelling and Analysis of Timed Systems, LNCS 4763, pages 211-226, Springer-Verlag, 2007.

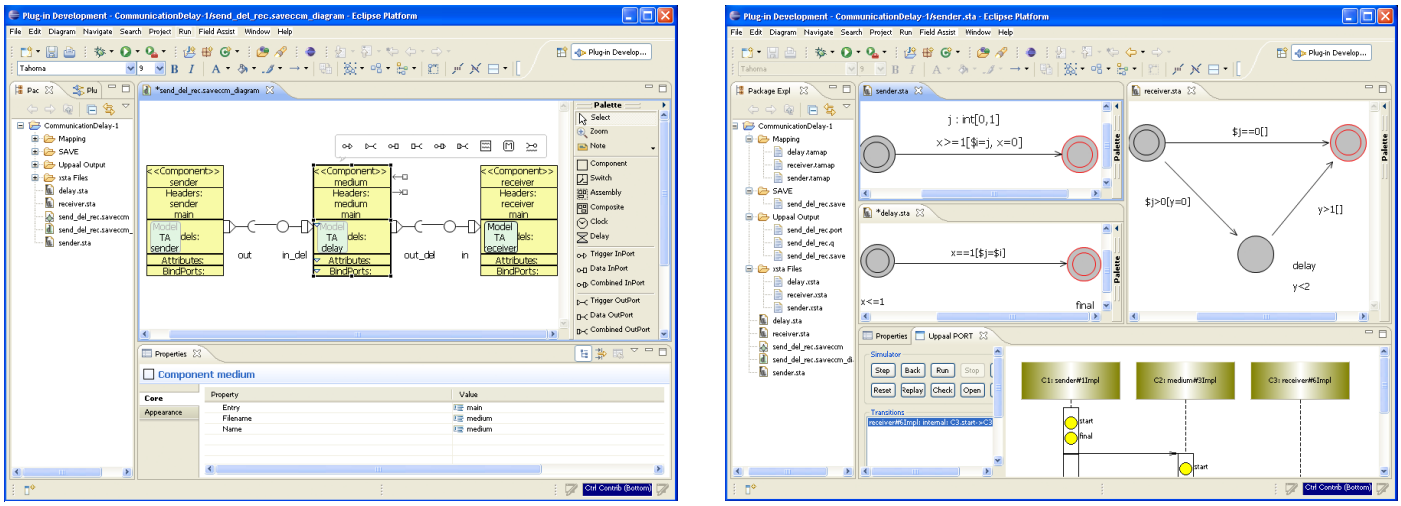

Figure 1. Save-IDE screenshots – the architecture editor (left screenshot), the timed automata editor (upper part of the right screenshot), and the behavior simulator (lower part of the right screenshot).

[2] UPPAAL PORT is available for download from the webpage http://www.uppaal.org/port