Save-IDE – A Tool for Design, Analysis and Implementation of Component-Based Embedded Systems

Séverine Sentilles, Anders Pettersson, Dag Nyström,
Thomas Nolte, Paul Pettersson, Ivica Crnkovic
Mälardalen Real-Time Research Center, Mälardalen University, Västerås Sweden
{severine.sentilles, anders.pettersson, dag.nystrom,
thomas.nolte, paul.pettersson, ivica.crnkovic}@mdh.se

Abstract

The paper presents Save-IDE, an Integrated Development Environment for the development of component-based embedded systems. Save-IDE supports efficient development of dependable embedded systems by providing tools for design of embedded software systems using a dedicated component model, formal specification and analysis of component and system behaviors already in early development phases, and a fully automated transformation of the system of components into an executable image.

1. Introduction

Certain domains such as dependable embedded systems require having a high-confidence in the quality of products being developed. For this, a fundamental desiderata is to have the ability to deal with requirements such as dependability (e.g. reliability, availability, safety), timing (such as release and response time, execution time, deadline), and resource utilization (including memory, CPU, message channels, power consumption). This demands a strong emphasis on the analyzability and automation of the development process to ensure the necessary quality of the final products with respect to these requirements.

At the same time the growing complexity of embedded systems requires methods that increase the abstraction level, improve reusability, and enable concurrency in the development process. An approach to achieve this is Component-based software engineering (CBSE). Both types of requirements (development efficiency, and dependability) can be achieved using the component-based development approach based upon formally analyzable component-models and complemented with adequate analysis tools. However, most component-based technologies today lack the formal analysis tools needed to ensure dependability.

In this paper we present the *Save Integrated Development Environment* (Save-IDE) which gathers tools and techniques needed in the development process of dependable embedded systems and integrates them with component-based development. It includes development support based on a component model SaveCCM [1] that is designed to enable efficient design of embedded systems and behavioral, temporal analysis of the model. Compared to the majority of existing IDEs which focus mainly on the programming aspect, the Save-IDE applies a novel approach which integrates the following activities: (i) design, (ii) analysis, (iii) transformations, (iv) verification and (v) synthesis. The paper briefly describes these development phases and the tools integrated into Save-IDE.

The rest of the paper is organized as follows. Section 2 gives an overview of the development process and Save-IDE. Sections 3, 4 and 5 describe the particular development phases and the supporting tool, namely component-based design, component and system analysis, and synthesis. Section 6 concludes the paper.

2. Software Development Process

The development process (designated SaveCCT - SaveComp Component technology) is designed as a top-down approach with an emphasis on reusability. It includes three mayor phases: Design, Analysis and Realization, as illustrated on Figure 1.

The process begins with the *system design* phase in which the system is broken down into subsystems and components compliant with the SaveCCM Component Model [2]. If components (partially) matching the requirements already exist, the *select and adapt* activity is taken. Otherwise, new component(s) need to be developed (i.e. the *component development* activity is taken). Correspondingly, the components are first analyzed and verified individually towards the requirements (*formal component verification*). In a following phase, after having reconstructed the system (or parts of the system) out of individual components and their assemblies (*system composition*), the obtained compositions also need to be analyzed and verified

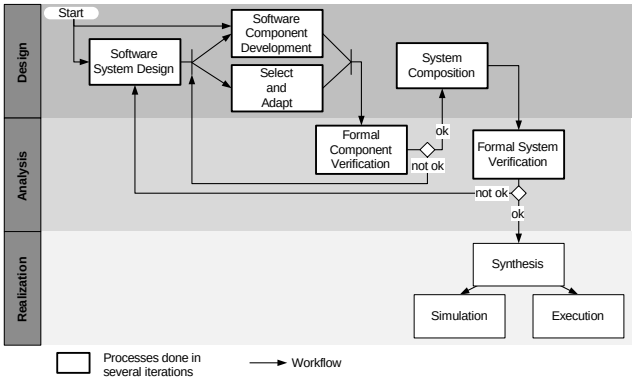

Figure 1. The SaveCCT development process

(*formal system verification*). The system and component design and verification procedure is being repeated until the results are acceptable from the analysis point of view. The phase that follows, the *realization* phase, consists of *synthesis* and *execution* or *simulation* activities. The system is synthesized automatically based on the input from the system design, on the implementations of the components and, on static algorithms for the resource usage and timing constraints. All the necessary glue code for the run-time system is produced. The resulted image can then be tested on a simulator or downloaded into the target platform.

The development process is semi-automatic, with several automated activities. A first automated activity is the production of the skeleton of the implementation files (C files and their corresponding header files) based on the specification of the component. Another one is the generation of the interchange file used as communication medium between tools [2]. The third one occurs during the synthesis which includes transformation of components into the executable real-time units, tasks, glue code generation, inclusion of a particular scheduling algorithm, compilation and linking all elements in the executable image.

This process is supported by a set of tools integrated into an Integrated Development Environment, Save-IDE [1]. The Save-IDE is designed as a platform with an extensible set of tools providing integrated support to achieve the SaveCCT approach as presented in [1, 8]. Save-IDE is developed as a set of plugins for the Eclipse framework and it comprises three key activities in the development process: (i) system and component development that includes modeling and design of the components, the architectural design of the system and specification and implementation of components, (ii) time analysis of the system and the components, and (iii) the synthesis that includes transformation from components to tasks, setup of execution parameters like priorities and periodicity of execution, glue code generation and compilation. Save-IDE enables interactive and automatic use of these tools and combines the entire development chain into a common environment.

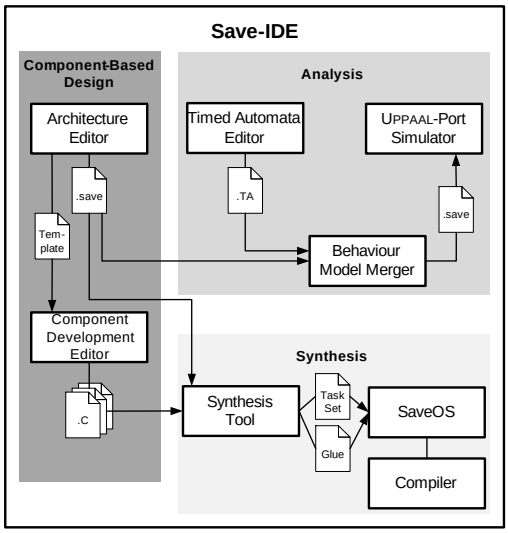

Figure 2. Overview of the Save-IDE tool-chain

In Figure 2, the organization of the Save-IDE tool-chain is shown. The development part consists of an *Architecture Editor* where system and component models can be created. Individual components can be implemented from generated c-template files in the C environment tool (CDT Eclipse plugin). In addition to the specification of functional interface, the Architecture Editor makes it possible to assign different attributes to the components, such as execution time, or behavioral model; for the latter the UPPAAL tool [6] and its front-end tool UPPAAL PORT [2] is used. Finally, systems can be synthesized using the synthesis tool. This process is done automatically. Synthesis is performed towards the SaveOS (Save Operating System), which is an abstraction layer that allows Save-based systems to be easily ported to different operating systems and hardware platforms. The final step in the chain is to compile and download the application to the target. Furthermore, using an external tool, CC-Simtech [4], systems can be simulated on a standard desktop computer.

3. Component-Based Design

As depicted in Figure 1, the design of a system in SaveCCT distinguishes between two independent activities: *software system design* and *software component development*. Software system design consists of designing a system out of independent and possibly already implemented components, i.e. components being produced through the component development activity.

[1]The Save-IDE is available for download from the web page http://sourceforge.net/projects/save-ide/

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

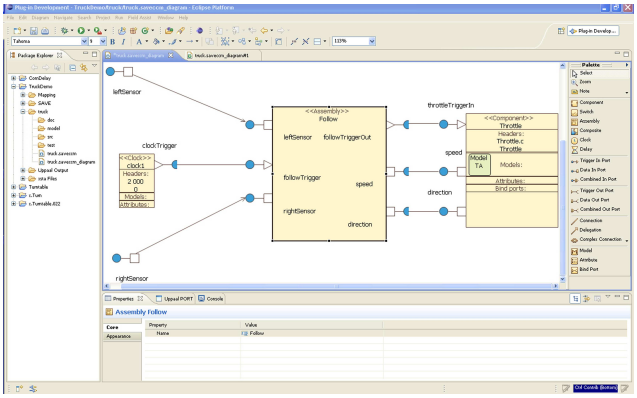

Figure 3. Architecture Editor

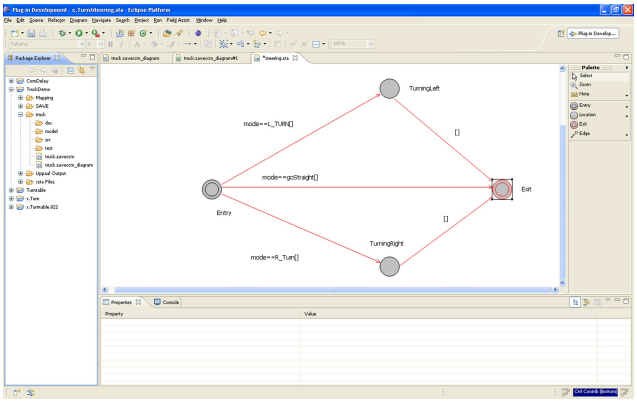

Figure 4. Behavioral Editor

The *Architecture Editor* enables designing a system following the semantics prescribed by the SaveCCM component model. To achieve tractable analysis of the system being developed (Section 4), the specification capability of this component model has been restricted. It consists of a minimum set of architectural elements (component, assembly, composite, clock, delay and switch) connected through "pipe-and-filter" ports distinguishing between control- and data-flows. Also the execution semantics of the components and composites (compound components) have been restricted to "read-execute-write" sequences performing computation (i.e. being active) when they are triggered by control ports. Otherwise, the components are in a passive state. More details about the component model can be found in [1] and [2].

For each composite architectural element two views coexist in the Architecture Editor (see Figure 3): the *external view* and the *internal view*. The external view describes the name and type of the element, the ports, and the models annotated to the element (such as time behavior represent by a timed automata), whereas the internal view handles the inner elements and their connections. This view can be hierarchical since SaveCCM allows hierarchical compositions of components and assemblies. The internal view presents the component implementation using the Component Development Editor provided by the Eclipse C/C++ Development Tooling (CDT). Skeletons for the C and header files containing mapping from ports to variables, function headers are generated by the Architecture Editor.

4. Analysis

The Analysis part in the Save-IDE consists of a Timed Automata Editor (TAE), a simulator, and a model-checker. The TAE provides the developer with a graphical user interface for creating a formal model of the internal behavior of a SaveCCM element. The behavior is described as a timed automaton [3] but with a distinct end location. The model of timed automata (TA) and its cost extended version priced timed automata is suitable for modeling functional and timing properties, and well as extra functional properties such as e.g., resource consumption.

Informally, the TA is assumed to start in its initial location when the element is triggered. The element then behaves as specified by the TA until it reaches its end location. At this point values are written to the output ports and the output trigger of the element is activated. Using a semiautomatic mapping process the user associates the external ports of a SaveCCM element with variables of the internal TA. In this way, it becomes possible to create formal models of individual elements composed into composite components or whole architectural descriptions.

The output of the TAE and the associated mapping can be compiled (by Save-IDE) into an XML-format accepted by the tool UPPAAL PORT which features a graphical simulator and a formal verifier. Using the simulator — which is graphically fully integrated into the Save-IDE — it is possible explore the dynamic behavior of a complete SaveCCM design in the early development phases of a project, prior to implementation. In this way, the designer can validate the design and gain increased confidence in the design. Using the verification interface, it is possible to establish by model-checking whether a SaveCCM model satisfies formal requirements specified as formulas in a subset of the logic Timed CTL. In this way, it is possible to achieve further increased confidence in the component-based design, w.r.t., e.g., functionality and timing.

The tool UPPAAL PORT is based on the timed automata model-checker UPPAAL [6], but extended with partial order reduction techniques which exploits the structure and semantics of SaveCCM model to improve the model-checking performance [5]. The technique and tool have been proven efficient for benchmark examples [5] and for an industrial control system [10].

5. Synthesis

As part of the Save-IDE tool chain, the synthesis includes a set of automated generation tools which transform and compile a SaveCCM-model allowing the developer to follow the SaveCCT work-flow in a more intuitive way. Via the graphical user interface the developer can invoke the tool chain by a simple mouse-click which invokes a sequence of tools.

There are three steps in the automated generation tool chain: *generation*, *synthesis* and *run-time environment compilation*.

The first step, generation, is a transformation of the model into auxiliary files in XML-format conforming to the SaveCMM-Language [2]. During the generation step the user creates template source files for each component in which the behavior of the component can be implemented.

The second step in the automated generation tool chain is the synthesis part, where the application is transformed from the component model into the execution model. The synthesis takes the SaveCCM model and constructs a set of trees based on the applications triggers. These trees are then used to generate the software code realized into the tasks, i.e., the function calls to the software components as well as glue code needed for passing data between the components. Each tree is mapped to one real-time task, and the configuration of the task is done with respect to the parameters of the trigger, e.g., setting of periods and priorities.

Finally, once the synthesis is performed, the run-time environment compilation and linking can be performed, and finally the executable can be downloaded on the hardware target or executed by a simulator.

The synthesis is independent of the run-time environment by the use of SaveOS, an abstraction layer between the actual run-time environment and the application. The applications do not call any native operating system services directly, but indirectly calling services using SaveOS application programming interface. SaveOS is designed and implemented in a way that it requires minimal computing and memory resources and provides a neglecting overhead. By using the SaveOS the configuration of the run-time environment can be changed without having to change the model or the implemented behavior of the components.

6. Conclusion

We have presented the Save-IDE, an integrated development environment that provides support in the development of predictable component-based embedded systems following the approach which emphasizes on formal behavior modeling and automated generation of the executable. As future work we plan to extend the modeling language to a richer component model, called ProCom [9], and a new language, called REMES [7], for modeling of internal and external component behaviors and embedded resources.

References

[1] M. Åkerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Håkansson, A. Möller, P. Pettersson, and M. Tivoli. The SAVE Approach to Component-Based Development of Vehicular Systems. *Journal of Systems and Software*, 80(5):655–667, May 2007.

[2] M. Åkerholm, J. Carlson, J. Håkansson, H. Hansson, M. Nolin, T. Nolte, and P. Pettersson. The SaveCCM Language Reference Manual. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-207/2007-1-SE, Mälardalen University, January 2007.

[3] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] CC Systems AB. CCSimTech. http://www.cc-systems.com/.

[5] J. Håkansson and P. Pettersson. Partial Order Reduction for Verification of Real-Time Components. In J.-F. Raskin and P. Thiagarajan, editors, *Proceedings of the 5th International Conference on Formal Modelling and Analysis of Timed Systems, Lecture Notes in Computer Science 4763*, pages 211–226. Springer Verlag, October 2007.

[6] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.

[7] C. Seceleanu, A. Vulgarakis, and P. Pettersson. REMES: A Resource Model for Embedded Systems. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-232/2008-1-SE, Mälardalen University, October 2008.

[8] S. Sentilles, J. Håkansson, P. Pettersson, and I. Crnkovic. Save-IDE – An Integrated Development Environment for Building Predictable Component-Based Embedded Systems. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*, September 2008.

[9] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic. A Component Model for Control-Intensive Distributed Embedded Systems. In M. R. Chaudron and C. Szyperski, editors, *Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE2008)*, pages 310–317. Springer Berlin, October 2008.

[10] D. Slutej, J. Håkansson, J. Suryadevara, C. Seceleanu, and P. Pettersson. Analyzing a Pattern-Based Model of a Real-Time Turntable System. In *6th International Workshop on Formal Engineering approaches to Software Components and Architectures(FESCA), ETAPS 2009, York, UK*. Electronic Notes in Theoretical Computer Science (ENTCS), Elsevier, March 2009.