COMDES Development Toolset

Guo, Yu; Sierszecki, Krzysztof; Angelov, Christo K.

Published in:
Proc. of the 5th International Workshop on Formal Aspects of Component Software FACS'2008

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

COMDES Development Toolset

Yu Guo [1] Krzysztof Sierszecki [2] Christo Angelov [3]

Mads Clausen Institute
University of Southern Denmark
Soenderborg, Denmark

Abstract

This paper presents the COMDES Development Toolset, which supports component-based development of embedded software in a model-driven fashion. The tool, which is built on the Eclipse platform, provides facilities for the graphical specification of a COMDES application. Once the application is specified, the application model is used as input to a transformation engine that generates the final configuration model involving pre-defined component models that are retrieved from a component repository. This model is finally used to generate source codes and executable, which is deployed in the target system.

Keywords: component-based design, model-driven development, metamodeling, Eclipse, tools

1 Introduction

Traditional methods used to develop embedded software are plagued by a number of problems, e.g. language incompatibility between the application and IT domains, and discontinuity gaps arising in the process of development, whereby the implementation is not always consistent with the specification.

The first problem can be solved by means of domain-specific modeling techniques (languages), which bridge the application domains and the IT domain. The COMDES framework [5] provides such a language for the embedded control systems domain. Model-driven software development [2][3] seems promising to solve the second problem, since the implementation can be derived from, or generated directly from the specification. However, the outlined approach requires adequate tool support.

The COMDES Development Toolset is intended to give a solution of above problem via a modeling framework based on reusable components with robust, flexible and extensible tool support. This paper provides a brief introduction to COMDES and presents its software development toolset. The rest of the paper is structured as

[1] Email: guo@mci.sdu.dk
[2] Email: ksi@mci.sdu.dk
[3] Email: angelov@mci.sdu.dk

follows: Section 2 presents the COMDES framework and its components. Section 3 deals with the architecture and implementation of the toolset. A summary is given in the concluding section of the paper.

2 COMDES Framework

The COMDES framework provides a domain-specific language that enables modeling and assembling of concrete applications using prefabricated executable components that are typical for the embedded control systems domain. System structure is specified statically in terms of distributed embedded actors that communicate with each other by exchanging labeled messages (signals), see Fig. 1a. Signal-based communication provides for transparent interaction between actors, independent of their allocation on network nodes. Actors (see Fig. 1b) are built from reconfigurable components – function blocks (FBs), and their structure is specified with FB diagrams, see Fig. 1c.

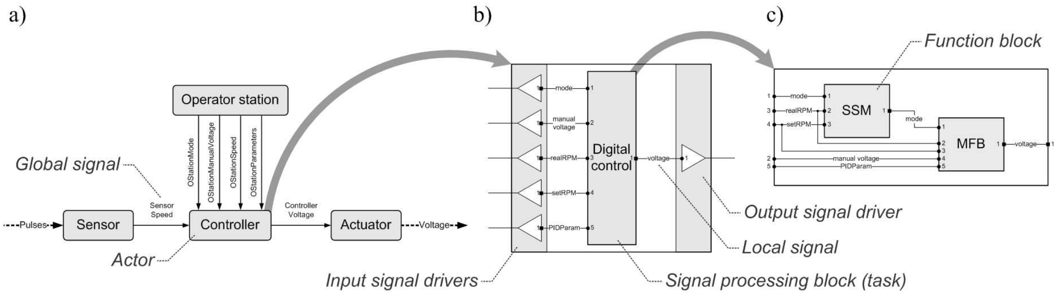

Fig. 1. COMDES system architecture

A COMDES function block has three aspects: kind, type and instance. COMDES defines five kinds of function block: basic, composite, signal driver, modal and state machine. A basic FB is specified by subsets of inputs and outputs, as well as a number of functions transforming input signals into output signals. A subset of component instances may be softwired into a software circuit that can be encapsulated into a composite FB, whereby constituent instances are executed in a sequence that is determined by the flow of signals in the function block diagram. Signal drivers are a special class of FB, which are wrappers providing an interface to the system environment by executing functions that are hardware or kernel-dependent. State machines are defined by a state chart, which is used to determine the current execution state based on a set of Boolean inputs. State machines operate in conjunction with modal FBs. The latter contain a number of slots (modes) encapsulating other components. Only one slot can be executed at a time, which is determined by the state output of a state machine controlling the execution of the modal FB.

Function block types can be reused across multiple projects (e.g. filter, PID, etc.). They are instances of kind and can be instantiated when building the application. Types are stored in a component repository and define the behavior of a component. When developers are building the application, they load the component types from a repository and instantiate them. Instances of the same type share the same behavior but differ by the data structure. Typically, there are many instances of a component of a given type.

Component types are created by skilled software engineers in the IT domain using specific design patterns. In contrast, an application is configured by control domain experts from already available components found in the repository. The creation of the components and applications is both supported by the COMDES development toolset.

3 The Architecture of the COMDES Toolset

Software systems are not static and are liable to significant changes, particularly during the first phases of their lifecycle. By using the Model-Driven Software Development approach, models and codes are better integrated in the sense that the changes of the application are effected through the models rather than modifying the code directly. The COMDES framework employs patterns of C code for each kind of component, which are derived from the corresponding component models. Thus, it is possible to automatically convert models to code and ultimately – generate the application.

The COMDES software development tools and their relationship are shown in Fig. 2. These include *Editor*, *Configurator* and *Generator* implemented as Eclipse plug-ins. The Compiler and Linker are hardware-specific being dependent on the microcontroller used by the application. Means of accessing the Repository are provided for all tools. Various models are exchanged among tools, and the final output is the executable code for a specific platform.

The underlying idea is that models (gray boxes) are used in each step throughout the entire development process; therefore, each step of the process is related to some kind of model transformation, e.g. transformation from specification to configuration model, and from configuration model to code.

The specification model of an application is created by developers using the graphical Editor. The model specifies the application in terms of COMDES components. This model is then processed by the Configurator to be extended with the components that the application is dependent on. The configuration model produced by the Configurator contains all information necessary for the Generator to output *glue* codes. It is important to note that these are data structures representing component instances, which define the application configuration by gluing components together. No executable code is generated. Instead, the glue codes are compiled and linked with component types, stored in a repository as pre-compiled objects, to form the final executable on the target. Currently, the COMDES components are implemented in C that is also used as the 'de facto' language of embedded systems. Therefore, existing compilers and linkers provided by the GNU tool chains can be employed to obtain the executable.

As discussed in the previous section, the component models are stored in the repository as types. During the design stage, component instantiation is realized by creating a reference from an instance to a type. In the process of specifying an application, the developers create an instance object without any type. The instance should be subsequently referred to a type, which has been loaded onto the devel-

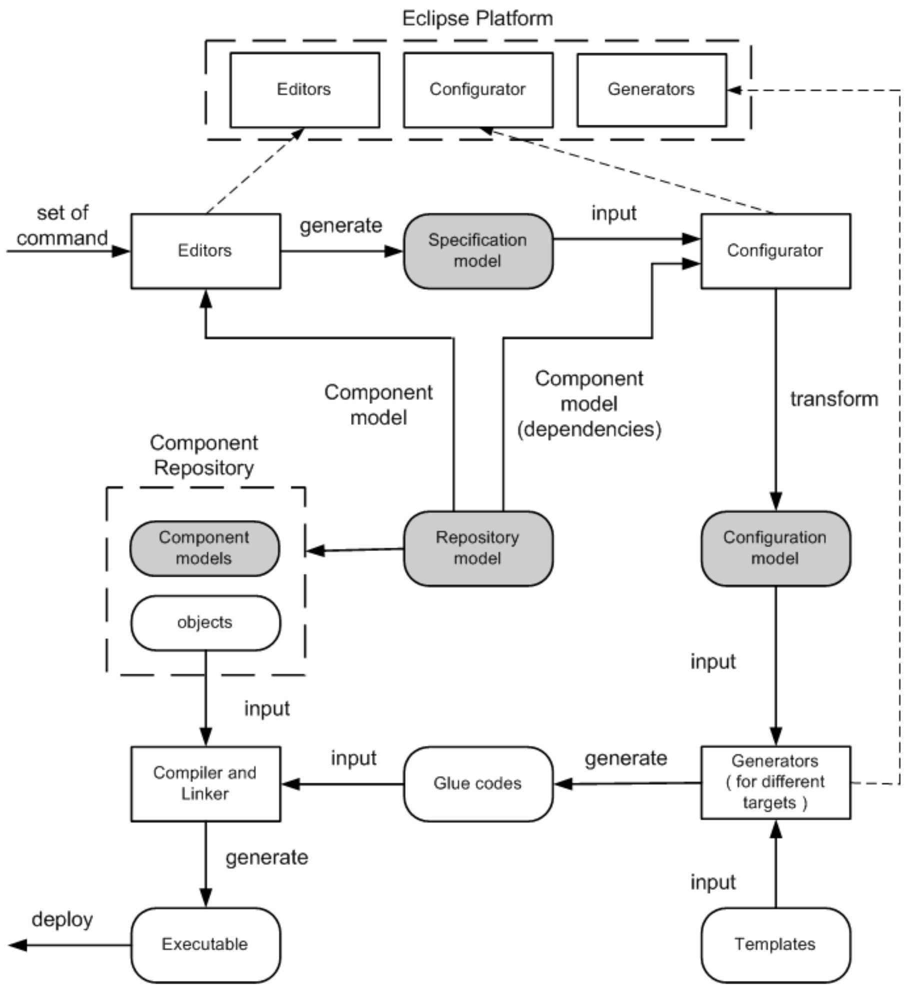

Fig. 2. The architecture of the COMDES Development Toolset

opment environment from the repository through a repository model that stores the links to all components. This approach of loading components from the repository is flexible in that it can be implemented in any object-oriented environment. It avoids the requirement for the tool implementation environment to support runtime definition (construction) of classes [4].

The Editor generates specification models, which consist of components and connections between them. The metamodel of the specification model comes from a collection of concepts within the specific application domain. It is platform-independent and does not provide knowledge about the real implementation. In contrast, the configuration models are simple flat models used to generate code. Therefore, the corresponding metamodel is an abstraction from the COMDES component design pattern. It is platform-dependent since the C libraries are used for the real implementation. It contains all information needed to generate the code.

The Configurator performs the transformation from the specification model to configuration model [1]. The latter is used by the Generator to produce glue codes with the support of templates. No manual coding is necessary, as the application-specific logic has been defined as component types, which are pre-defined and stored in the repository in the form of objects. The Linker links these objects to generate executable code. Different component instances of the same type differ only in data, whereas the routine that processes the data is the same for all instances of a given type.

The toolset is implemented in Eclipse (http://eclipse.org), in order to reduce the amount of manual work needed with conventional methods. The COMDES metamodel is built in the Ecore language of the Eclipse Modeling Framework (EMF). The graphical editing facility is provided by the Eclipse Graphical Modeling Framework (GMF). Once the metamodel is built, the EMF can generate a set of Java implementation classes conforming to the metamodel. These classes can be easily used to parse the model, which allows for transformations between models. Alternatively, a model transformation language, like the Atlas Transformation Language (ATL), can also be considered to perform the transformation task. Generation of the glue codes is carried out by the CodeWorker (http://codeworker.free.fr) since it provides an expressive scripting language to write templates that guide the generation process. Moreover, CodeWorker provides a scripting language adapted to the description of any input format, in our case the XML Ecore files.

4 Conclusion

The paper has presented the COMDES Development Toolset, which has been specifically designed to support embedded control system development. The latter is based on the COMDES framework, which provides a component-based language for control domain applications. Being a graphical language, COMDES facilitates application development – the developer uses a specialized editing environment to specify the application, making COMDES intuitive and easy to use, without the need of extensive training.

An initial prototype of the toolset has been developed on the Eclipse platform. It provides facilities for creating, editing, transformation and generation of application and component models. The control application can be specified graphically and the final executable code of the application can be created by generating and gluing together instances of executable components stored in a component repository. The development of an improved version of the toolset is currently in progress.

References

[1] Guo,Y, K. Sierszecki and C. Angelov, *A (Re)Configuration Mechanism for Resource-Constrained Embedded Systems*, Proc. of the 1st IEEE International Workshop on Component-Based Design of Resource-Constrained Systems, Turku, Finland, July/Aug 2008

[2] Object Management Group, *MDA Guide Version 1.0.1*, 12th June 2003

[3] Stahl,T., M. Völter, J. Bettin, A. Haase, S. Helsen, K. Czarnecki (Foreword by), B. von Stockfleth (Translated by), "Model-Driven Software Development: Technology, Engineering, Management", ISBN: 978-0-470-02570-3, Wiley, 2006

[4] Thramboulidis, K., G. Doukas, A. Frantzis, *Towards an implementation model for FB-based reconfigurable distributed control applications*, Proc. of Seventh IEEE International Symposium on Object-oriented Real-time Distributed Computing, Vienna, Austria 2004

[5] Xu, Ke, K. Sierszecki and C. Angelov, *COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems*, Proc. of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Daegu, Korea, 2007