Science of Computer Programming 97 (2015) 17–22

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

The role of supervisory controller synthesis in automatic control software development

Jos Baeten[a,b,1], Jasen Markovski[b,*,1]

[a] *Centrum Wiskunde & Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*
[b] *Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

HIGHLIGHTS

- We discuss the role of supervisor synthesis in automated software code generation.
- The proposed approach is systematic and based on process theory.
- We implemented a model-based systems and software engineering framework.
- The framework has been applied to multiple industrial studies.

ARTICLE INFO

Article history:
Received 26 September 2013
Accepted 5 November 2013
Available online 21 November 2013

This article is dedicated to Paul Klint, an exemplary computer scientist, who always manages to combine sound theoretical foundations and practical implementations in software

Keywords:
Supervisory control theory
Model-driven development
Software synthesis

ABSTRACT

We give an overview of a model-driven systems engineering approach for high-tech systems that relies on supervisory controller synthesis. The proposed framework has a process-theoretic foundation and supports extensions with quantitative features. We briefly discuss several industrial case studies that highlight the advantages of the proposed approach.

1. Model-driven control software development

Embedding information and communication technology in consumer and industrial products was enabled by advances in software that carries most of these products' functionalities. The need for development techniques that can guarantee software quality is more than apparent. Conferences like Software Development Automation, Model-Driven Engineering Languages and Systems, or Applications of Concurrency in System Design, are some of the venues where this need has been recognized and studied by means of automated model-driven software development techniques. In this paper, we would like to point to one model-driven systems engineering approach, referred to as supervisory controller synthesis, which targets discrete-event control software for high-tech and complex systems. We find this approach to be relevant to the software development community and we hope that it might offer novel insights in development of quality control software, and bring the communities of software development, systems engineering, and formal methods closer together.

* Corresponding author.
E-mail addresses: jos.baeten@cwi.nl (J.C.M. Baeten), j.markovski@tue.nl (J. Markovski).
[1] Supported by Dutch NWO project: ProThOS, no. 600.065.120.11N124.

0167-6423/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.scico.2013.11.016

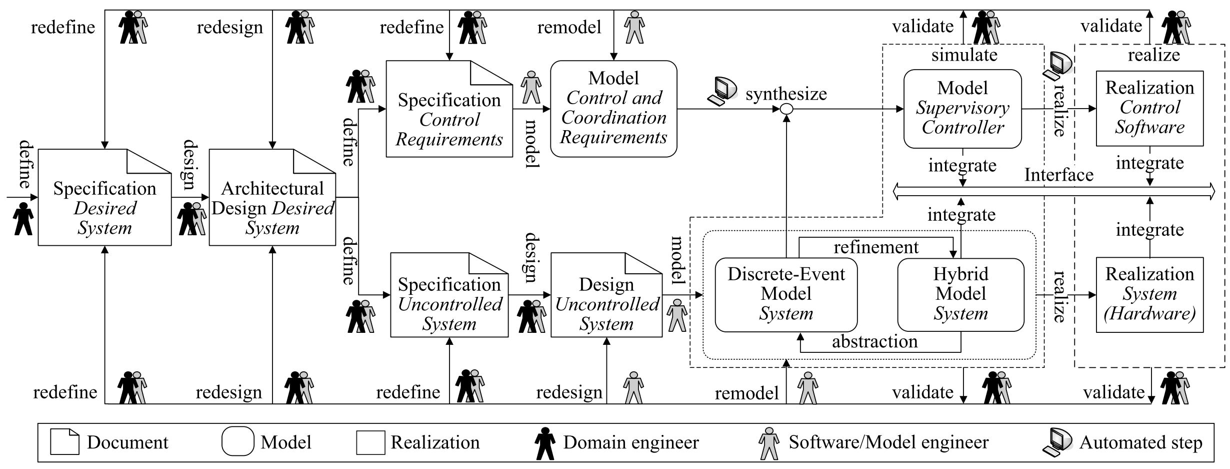

Fig. 1. A synthesis-centric model-driven systems engineering framework.

Development of quality control software is becoming an important challenge in the production process of high-tech complex systems. The ever-increasing complexity, in conjunction with the market pressure for improved quality, safety, ease of use, and performance, promote the former as an important future bottleneck. Many model-driven software development approaches, typically relying on formal methods, are advocated as cost-effective approaches that support rigorous specification, validation, and testing of software [1,2]. Some standards, like [3,4], even make these methodologies compulsory. Admittedly, the use of formal methods requires a substantial effort in the initial development and design phases. However, formal specifications and early model validation greatly decrease the number of errors found during testing and integration phases mitigating overall product-development time and costs, while increasing quality [5].

Unfortunately, there is still insufficient emphasis on the integration of the formal methods in the systems engineering process, e.g., as noted in [6], despite successful industrial cases involving formal verification and validation like [7–9]. Furthermore, it has been noted early on [10] that traditional software development approaches are not entirely adequate to control software development. This is mainly due to the fact that control and coordination requirements, which typically specify safety properties in an informal and often ambiguous manner, frequently change during the design process, inducing a large number of expensive (re)coding–validation–testing iterations. Consequently, partially employing formal techniques for some development phases without supporting the complete design process might not suffice, calling forth a shift from process-based towards model-driven development [11].

2. A synthesis-centric approach

To mitigate some of the above issues in the design and development of supervisory control software for high-tech complex machines, we propose to employ a synthesis-centric model-driven systems engineering framework developed in [12–14], depicted in Fig. 1. Supervisory controllers observe and coordinate the discrete-event behavior of the concurrently running components of the system. Based on the observations made from the uncontrolled system, these controllers make a decision on which activities the system is allowed to perform, and send back control signals that actuate the system. The layer of supervisory control is on a high level of abstraction, residing between the user interaction and the resource (embedded) control of the machine [15]. The framework is synthesis-centric as models of the supervisory controllers are synthesized automatically based on the models of the uncontrolled system and the control requirements.

The modeling process is initiated by domain engineers that propose an informal specification of the desired system. Subsequently, an architectural design of the system is made in cooperation with software engineers, which most importantly defines the control software architecture and modeling level of abstraction. Based on the design, specification and models of the system and the control and coordination requirements are made in parallel. The model of the system comprises the models of its (synchronizing) components, whereas the control requirements specify the allowed (safe) behavior of the system.

Most systems contain mixed continuous and discrete-event or hybrid behavior and hybrid models of the system are made for the purpose of simulation and validation. These models are abstracted to a discrete-event model, required by the synthesis procedure [16,15,17]. The discrete-event model of the uncontrolled system together with the model of the control requirements serve as input for the synthesis of the model of the supervisory controller. The control software is automatically generated based on the synthesized model of the supervisory controller. The framework provides for software and hardware-in-the-loop simulation for validation of the control against the models or the prototypes of the system, respectively.