Mechatronics xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems

Marco Aurélio Wehrmeister [a,*], Edison Pignaton de Freitas [b], Alécio Pedro Delazari Binotto [c], Carlos Eduardo Pereira [d]

[a] Federal University of Technology – Paraná (UTFPR), Av. Sete de Setembro 3165, 80230-901 Curitiba, Brazil
[b] Federal University of Santa Maria (UFSM), CESNORS, 98400-000 Frederico Westphalen, Brazil
[c] IBM Research – Brazil, Rua Tutóia 1157, 04007-900 São Paulo, Brazil
[d] Electrical Engineering Department (DELET), Federal University of Rio Grande do Sul (UFRGS), Av. Osvaldo Aranha 103, 90035-190 Porto Alegre, Brazil

ARTICLE INFO

Article history:
Received 30 December 2012
Revised 21 December 2013
Accepted 23 December 2013
Available online xxxx

Keywords:
Model-Driven Engineering (MDE)
Aspect Oriented Software Development (AOSD)
Embedded and real-time system
Industrial mechatronics system
Design automation
Code generation

ABSTRACT

Recent advances in technology enable the creation of complex industrial systems comprising mechanical, electrical, and logical – software – components. It is clear that new project techniques are demanded to support the design of such systems. At design phase, it is extremely important to raise abstraction level in earlier stages of product development in order to deal with such a complexity in an efficient way. This paper discusses Model Driven Engineering (MDE) applied to design industrial mechatronics systems. An aspect-oriented MDE approach is presented by means of a real-world case study, comprising requirements engineering up to code generation. An assessment of two well-known high-level paradigms, namely Aspect- and Object-Oriented paradigms, is deeply presented. Their concepts are applied at every design step of an embedded and real-time mechatronics system, specifically for controlling a product assembler industrial cell. The handling of functional and non-functional requirements (at modeling level) using aspects and objects is further emphasized. Both designs are compared using a set of software engineering metrics, which were adapted to be applied at modeling level. Particularly, the achieved results show the suitability of each paradigm for the system specification in terms of reusability quality of model elements. Focused on the generated code for each case study, statistics depicted an improvement in number of lines using aspects.

1. Introduction

Modern automation and control systems include electro-mechanical devices controlled by complex embedded and real-time systems, comprising hardware and software components. These systems enable the creation of "smart" or "intelligent" automation devices which are able to execute autonomously and support fully decentralized decision making. This dramatically changes the architectures (usually centered on Programmable Logic Controllers – PLC) adopted in industrial automation.

The increasing number of functionalities incorporated into these modern embedded and real-time systems may require not only specialized hardware and software components, but also their deployment over different processing units, being possibly physically separated. Real-time constraints affect both processing and communication. Handling such requirements cannot violate other system constraints and/or requirements. Therefore, engineers must deal not only with the design of software and hardware in the same project, but also with their interaction which is generally implemented via industrial communication protocols [37,14]. In this sense, it is important to minimize (or even avoid) inconsistencies in system specification, i.e., software and hardware teams must follow the same consistent system specification basis.

Furthermore, the non-functional nature of some important requirements of embedded systems can lead to several problems, such as scattered and tangled handling. If they are not properly handled, these problems increase the overall design complexity, affecting effort and project timeline. In this case, reuse of previously developed artifacts (e.g., software and/or hardware blocks) becomes harder. Additionally, software and hardware components are usually designed concurrently and using distinct languages, tools, and concepts, considerably increasing design complexity.

Several works propose the raising of abstraction level and separation of concerns in order to manage the growing complexity. Some works propose the use of high-level concepts from *Object-Oriented* (OO) paradigm [50,6]. However, to specify the handling

* Corresponding author. Tel.: +55 4133104745.
E-mail addresses: wehrmeister@utfpr.edu.br (M.A. Wehrmeister), edison.p.freitas@ufsm.br (E.P. de Freitas), abinotto@br.ibm.com (A.P.D. Binotto), cpereira@ece.ufrgs.br (C.E. Pereira).

0957-4158/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mechatronics.2013.12.008

of non-functional requirements using only the concepts available in OO paradigm is not adequate. OO paradigm lacks convenient abstractions to represent and encapsulate non-functional requirements handling. More precisely, non-functional requirements handling is scattered and intermixed within many objects responsible for handling functional requirements. To overcome such problems at implementation level, subject-oriented programming [35] and *Aspect-Oriented* programming [28] have been proposed. Both works promote a **partitioning of crosscutting concerns in units of modularization called subjects and aspects**, respectively.

Model-Based Engineering [38,21] and/or *Model-Driven Engineering* (MDE) [43,1] are approaches intended to raise abstraction level by using models as the main artifacts created during design. The main idea is to create a *Platform Independent Model* (PIM) which is refined via model transformations into a *Platform Specific Model* (PSM). MDE can be seen as a trend in designing embedded systems for automation applications and several approaches have been proposed in the last years, like [48,23,21,1,2].

It is important to highlight that the use of models during design is not a completely new proposal. Other engineering disciplines have been using models for decades. However, particularly for systems comprising hardware and software (which is the case for embedded systems), models can play a more active and important role than only project documentation. Models shall be used to (automatically) generate system implementation through model transformations, keeping specification and implementation synchronized. In addition, model transformations enable *correct-by-construction* implementations, provided that models and transformations are formally and semantically proved. Meanwhile, intermixing the handling of requirements from different natures at modeling level is still a challenge task.

Within this context, this paper discusses the use of MDE and separation of concerns for handling functional and non-functional requirements. This article presents in details an innovative approach called **AMoDE-RT – Aspect-oriented Model Driven Engineering for Real-Time systems**, which combines *Unified Modeling Language* (UML)[1] with concepts of *Aspect-Oriented Software Development* (AOSD) [22]. AMoDE-RT conceptualizes the separation of concerns with the handling of functional and non-functional requirements from earlier design stages (e.g., requirements engineering and modeling phases) to final implementation.

Practically, AMoDE-RT is presented throughout this text by means of a case study representing a real-world industrial mechatronics system, namely the control system for a product assembler industrial cell. Therefore, this paper contributes to the following goals: (i) apply AOSD concepts together with UML at modeling level; (ii) demonstrate the use of UML to model a concrete embedded and real-time system for controlling an industrial mechatronics system; (iii) assess both OO and AOSD in terms of UML models through software engineering metrics, comparing their strengths and weaknesses; and (iv) promote a discussion on the use of AOSD within design of embedded and real-time systems applied to mechatronics systems.

AMoDE-RT increases the reuse of artifacts produced during design, as it allows for a better separation of concerns in requirements handling. For that, the *Distributed Embedded Real-time Aspects Framework* (DERAF) has been developed to be used during the whole design process, from initial phases until system implementation. This framework provides a predefined set of aspects which deals with non-functional requirements commonly found in automation systems (see Section 5). Although there are crosscutting concerns related to functional requirements, this work concentrates on those associated with non-functional requirements. The presented results demonstrate indicators on the effectiveness of AMoDE-RT to design embedded and real-time systems for industrial automation applications.

This article is organized as follows: Section 2 discusses related work, followed by Section 3 that provides an overview of AMoDE-RT design flow. Section 4 introduces the case study used throughout this text. The requirements engineering process proposed in AMoDE-RT is described in Section 5. The specification of functional requirements using UML is discussed in Section 6 while non-functional requirements specification is approached in Section 7. Section 8 presents the tools created to support the AMoDE-RT approach. An assessment of AMoDE-RT and results obtained for the case study are presented in Section 9, which includes a discussion on reusability of design artifacts and also on applying MDE in industry. Finally, conclusions and future work directions are discussed in Section 10.

2. Related work

This section briefly discusses some relevant related work on applying MDE in the domain of automation and embedded systems. An interesting survey on using UML in mechatronics systems design is presented in [50]. The author identifies an increasing number of researchers proposing the use of UML as a specification language to complement traditional approaches, such as those using MATLAB, Simulink, and Statecharts.

Traceability is an important feature in MDE to enhance automated analysis, consistency, and coherence of models used during software development. The work of [36] contributed with a management approach for the complexity of traceability information in MDE by means of identifying trace-links in a MDE process and of defining semantically rich trace-links between models. Identifying rich trace-links is crucial to maintain traceability during software development.

Targeting software product line engineering, [17] shows important issues to include an aspect-oriented MDE tool-chain in the software development production line process. The *Aspect-oriented Model-driven Product Line Engineering* (AMPLE) project[2] proposed an aspect-oriented MDE methodology for Software Product Line (SPL), aiming to improve modularization of software variations and maintenance of their traceability during SPL evolution. The lifecycle proposed in AMPLE comprises early activities as requirements engineering, as well as architecture definition and implementation activities of a software based on SPL. In this sense, AMoDE-RT is similar in the way it uses AOSD and MDE techniques and covers a similar range of activities. However, *AMoDE-RT focuses not only on software but also on hardware components of an embedded system*. This is achieved by using a Platform-based Design approach, which comprises platforms that provide hardware and software components, and also model transformations and code generation. Nonetheless, AMoDE-RT is more restricted when using AOSD, since it applies aspects to deal with non-functional crosscutting concerns, whereas AMPLE deals with both functional and non-functional ones.

The work presented in [20] applies Theme/UML, an aspect-oriented MDE approach to separate embedded system concerns from earlier design phases to system implementation, reducing design complexity. That work illustrated the Theme/UML approach using a design of a pacemaker as case study, from modeling down to code. According to the authors, Theme/UML has some limitations: (i) it addresses only non-functional concerns that manifest themselves as code in the system; (ii) inheritance is not supported in UML-to-C transformation; and (iii) the behavior specified within aspects can be specified only with sequence diagram, leading to

[1] Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/.

[2] http://www.ample-project.net/.

Please cite this article in press as: Wehrmeister MA et al. Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems. Mechatronics (2014), http://dx.doi.org/10.1016/j.mechatronics.2013.12.008