King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A new method for constructing and reusing domain specific design patterns: Application to RT domain

Saoussen Rekhis [a,*], Nadia Bouassida [a], Rafik Bouaziz [a], Claude Duvallet [b], Bruno Sadeg [b]

[a] *MIRACL, Pôle Technologique de Sfax, BP 242, Sfax 3021, Tunisia*
[b] *LITIS, UFR des Sciences et Techniques, BP 540, 76 058 Le Havre Cedex, France*

Received 18 September 2015; revised 19 April 2016; accepted 20 April 2016
Available online 27 April 2016

KEYWORDS

Design pattern engineering;
Model transformation;
Real-time application modeling

Abstract Domain specific design patterns capture domain knowledge and provide solutions of non trivial design problems in a specific domain. Their application improves considerably the quality of software design. In order to benefit from these advantages and to reinforce the application of these patterns, we provide, in this paper, new processes and tools for the development and the instantiation of domain specific design patterns, especially those intended for real-time domain.

Initially, we propose a pattern development process that guides pattern developers in the construction of patterns. The proposed process defines unification rules that apply a set of comparison criteria on various applications in the pattern domain. This process is illustrated through the design of the controller pattern. Moreover, we propose a process guiding the application designers in pattern instantiation based on model transformation. Finally, the proposed RT patterns and their development process are evaluated by calculating quality metrics and comparing the applications designed with our RT patterns and others developed by experts without the use of our patterns.

1. Introduction

Design patterns (Gamma et al., 1994) represent solutions to common design problems in a given context. They improve substantially software quality and reduce the development cost. Nowadays, their use is wide spread since they capture and promote best practices in software design. However, the design patterns of GoF (Gamma et al., 1994) do not focus on a particular domain, thus they need a great adaptation effort since it is hard to determine in which context or in which part of the system these patterns can be used (Port, 1998). These reasons motivated several works on domain-specific patterns which encapsulate the essence of a certain domain

* Corresponding author.
E-mail addresses: saoussen.rekhis@fsegs.rnu.tn (S. Rekhis), nadia.bouassida@isimsf.rnu.tn (N. Bouassida), raf.bouaziz@fsegs.rnu.tn (R. Bouaziz), claude.duvallet@univ-lehavre.fr (C. Duvallet), bruno.sadeg@univ-lehavre.fr (B. Sadeg).
Peer review under responsibility of King Saud University.

(e.g., human computer interaction, security and real-time systems).

A domain-specific design pattern offers a flexible architecture with clear boundaries, in terms of well-defined and highly encapsulated parts that are in alignment with the natural constraints of the considered domain (Port, 1998). One of the domains where reuse through patterns will bring many benefits is the Real Time (RT) domain since it is a complex and evolving domain. The RT patterns help designers in developing applications that express time-constrained data and time-constrained methods. They provide various solutions to addressing the fundamental RT scheduling, communications and synchronization problems (Boukhelfa and Belala, 2015).

Design patterns specific to the RT domain are similar to any domain specific pattern; they need a representation language that shows the specificities of the domain. They need also a design process that helps in their construction and finally an instantiation process that will guide their reuse (Boukhelfa and Belala, 2015).

Nowadays, representation of domain-specific design patterns and their reuse receives special attention from many researchers (e.g., Kim et al., 2004; Kim, 2007). They proposed modeling languages in order to take into account pattern variability. In fact, when representing domain-specific patterns, the design language, e.g., UML has to support not only the flexibility characteristic of patterns, but also the specificities of the domain itself (Port, 1998). Nevertheless, in the example of real-time (RT) domain, the standard UML remains insufficient for expressing all features of RT applications. Therefore, different extensions to the UML language have been proposed to take into account RT application characteristics (Douglass, 2004; Lanusse et al., 1999; OMG, 2008). However, the proposed modeling languages are not suitable to patterns. That is, RT patterns must be generic designs intended to be specialized and reused by any application in RT domain. For this reason, in addition to the UML extensions representing RT aspects, we need new notations distinguishing the commonalities and differences between applications in the pattern domain.

On the other hand, the difficulty of the domain specific pattern development and specification slows their expansion. This is due essentially to the fact that they have to incorporate flexibility and variability in order to be instantiated for various applications in the domain. As a result there is a need for a design process that guides the domain-specific patterns development and defines rules to find similarities between a set of applications in the considered domain and their possible variations. This need is crucial in the RT domain since it is an evolving domain where the variety of applications is quite large and the reuse is very important. In RT domain, we distinguish the case where applications use a lot of RT data that must be stored in a database. We call this case "real-time domain with intensive data". Our contribution aims to guide the development of RT design patterns specific to RT domain with intensive data through the definition of unification rules that facilitate their specification.

Finally, note that, even when assisting the pattern developers in expressing and building design patterns, there is no certitude that these patterns would be correctly instantiated, by application designers. Thus, an ultimate assistance in validating the pattern instantiation would be of a valuable benefit. Several researchers (e.g., Kim and Carrington, 2006; Kajsa, 2013; Hammouda et al., 2009; Koskinen et al., 2010) were interested in the validation of pattern instances, however their approaches are not adapted to the RT domain. They do not take into account the UML extensions differentiating between passive resources and RT active resources and specifying RT features such as concurrency and deadline. As a consequence, there is still a need for an efficient guidance process for RT patterns reuse.

In our approach, we are interested in providing assistance, not only for pattern designers to support RT design pattern representation and development, but also for application designers to instantiate RT design patterns. This assistance is provided through:

(1) A UML profile intended for RT design pattern specification and reuse. This profile establishes a relation between the different systems that need similar facilities and provides a good ground to build on in order to establish a real pattern language for RT systems.
(2) A pattern development process using unification rules to determine the fundamental elements and the variation points of a pattern. This process facilitates the pattern developers' work, in order to specify patterns with a better quality. It addresses the structural and behavioral aspects describing how the different roles of a pattern interact.
(3) A guidance process for building and validating applications reusing patterns. When a pattern is deployed in an application design, some constraints must be preserved to make the pattern instantiation valid. The pattern instantiation process guides the application designers in patterns reuse and prevents them from violating pattern constraints.

The first contribution of this paper consists in showing how our pattern development process, initially presented in Rekhis et al. (2010), can be improved, fine-tuned and automated in order to define the structure of RT design patterns. Moreover, in this paper the process was augmented with the specification of the behavior of RT design patterns. A second contribution of this paper consists in proposing a new pattern instantiation process guiding the application designers in RT patterns reuse and preventing them from violating pattern constraints. We implement the instantiation process using an existing modeling framework, EMF, and incorporate the implementation as plug-in to the Eclipse modeling environment. The developed plug-in can interpret automatically the properties of the patterns since they are described in a precise manner using the UML-RTDP profile (Rekhis et al., 2013) that we defined previously. A third contribution of this paper consists in defining new metrics and using the CK metrics proposed in Chidamber and Kemerer (1994) to assess experimentally the efficiency of our design process. For the evaluation of the RT patterns obtained with our proposed development process, we propose some projects to experimented designers, while dividing them into two groups, one group models RT applications using our patterns approach, and the other will not use them. Then, we calculate some quality metrics for the produced projects (e.g., number of classes, average number of attributes, etc). The evaluation answers two questions: do the RT patterns have a good design quality? And do the RT patterns encapsulate really the concepts tied to RT domain?

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 presents the pattern