The Journal of Systems and Software 84 (2011) 1008–1021

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A framework for developing home automation systems: From requirements to code

Pedro Sánchez*, Manuel Jiménez, Francisca Rosique, Bárbara Álvarez, Andrés Iborra

Systems and Electronic Engineering Division (DSIE), Technical University of Cartagena, Campus Muralla del Mar s/n, 30202, Cartagena, Spain

ARTICLE INFO

Article history:
Received 28 June 2010
Received in revised form
14 December 2010
Accepted 19 January 2011
Available online 1 February 2011

Keywords:
Home automation
Model driven
Code generation

ABSTRACT

This article presents an integrated framework for the development of home automation systems following the model-driven approach. By executing model transformations the environment allows developers to generate executable code for specific platforms. The tools presented in this work help developers to model home automation systems by means of a domain specific language which is later transformed into code for home automation specific platforms. These transformations have been defined by means of graph grammars and template engines extended with traceability capabilities. Our framework also allows the models to be reused for different applications since a catalogue of requirements is provided. This framework enables the development of home automation applications with techniques for improving the quality of both the process and the models obtained. In order to evaluate the benefits of the approach, we conducted a survey among developers that used the framework. The analysis of the outcome of this survey shows which conditions should be fulfilled in order to increase reusability.

1. Introduction

Rapid advances in electronics, information and communications technology (leading to miniaturization and improvement of performance of computers, sensors and networking) have given rise to the development of several home automation (HA) technologies (Chana et al., 2009). HA applications integrate comfort, energy saving, security and communications functions. The aim of an HA system is to provide homes with a certain degree of 'intelligence' and to improve the quality of life of its inhabitants. Tasks like automatically switching lights and heating, cutting off the supply when gas or water leaks are detected or controlling the home devices remotely from a mobile or a computer through an Internet connection are typical applications of HA domain.

There are several HA standards and protocols adopted by the leading companies in the market. Some notable examples are KNX (ISO/IEC14543-3-X and EN50090 standards), Lonworks (ISO/IEC 14908, EN14908 and EIA-709-1 standards) and X10 (a well known international and open industry standard for communication among electronic devices). However, one of the main problems of HA development resides in the fact that there is no consensus in the standard to implement these applications. As stated in Miori et al. (2006), it is improbable that there will be a single dominant technology for HA in short term. Furthermore, each of such standards provides its own software suite to create HA applications and program the devices in question. Hence the particular technology (specific platform) must be selected at the initial design stage, as much as the tools and devices to be used depend on this choice. These facts make the development of HA applications strongly platform dependent, making it very difficult to raise the abstraction level and work with HA domain concepts rather than technology elements.

This drawback can be avoided by adopting the well known Model-Driven Development technique (MDD) (Selic, 2003). In this approach, application code can be automatically generated from platform-independent models. Although MDD techniques have been developed some years ago, there are no well known integrated frameworks for developing HA systems. However, there is currently a need for the creation of tools to develop these systems. These tools should allow the generation of code for several platforms. In this work, we present an integrated framework that allows the definition of HA systems at different levels of abstraction, from requirements to code. Taking advantage of using a domain specific language (DSL) (Mernik et al., 2005) the developer can work with graphical elements and concepts of the HA domain.

DSLs provide easy, intuitive domain-specific descriptions of systems using graphical or textual models. A DSL includes the tooling infrastructure for creating and transforming models into executable instances of the language (Kelly and Tolvanen, 2008). In this context, the appearance of the MDD approach has increased the research on these languages as well as new automatic code generation techniques. Nevertheless, the development of DSLs is very time

* Corresponding author. Tel.: +34 968326460; fax: +34 968325973.
E-mail address: pedro.sanchez@upct.es (P. Sánchez).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.01.052

consuming: ideally, models made using a particular DSL should be able to be reused across several implementations to amortize this effort. The reuse of DSL models across different development projects can help reduce the cost of these projects.

Several works discuss advantages and drawbacks of using DSLs. Maintenance, flexibility, productivity, reliability and reusability are attributes commonly found in these types of languages (Hermans et al., 2009). With DSLs, reuse is feasible at the model level, making it possible to reuse partial or entire models, rather than pieces of platform-dependent code. Thus, the beginning of a new software development project can be done from existing reusable assets. A surprising fact is that reuse hardly plays a significant role in current DSLs as demonstrated in Hermans et al. (2009) for the study of a particular DSL.

We identify two key aspects that determine the feasibility of reuse in the context of DSLs: (1) to select a model or a model fragment for reuse you must know what is does; and (2) to achieve effective reuse, you must be able to discover the model fragment faster than you could build it. Besides, the use of best practices for DSL definition and implementation determines the success in model reuse. For instance, language creators usually try to avoid modeling errors by imposing dozens of strongly enforced integrity rules that prevent modelers from temporarily breaking the rules while they are trying to reuse their models. Moreover, interconnected models should have minimal coupling to improve modularization and avoid data duplication which lead to maintenance and reuse problems.

In short, this article contributes to the state of the art with the following features:

- A framework that integrates a set of tools for defining HA applications at different levels of abstractions.
- A set of model transformations (Mens and van Gorp, 2006) that enables developers to get full executable code.
- Traceability capabilities (Ramesh and Jarke, 2001) to improve quality both of the process and of the models obtained.
- A survey that demonstrates the success of reusing models in MDD by means of generic requirements. We investigate factors that contribute to this success.

The article is structured as follows: Section 2 deals with introducing the basis of the proposal and the related works. Section 3 presents the proposed framework and also offers a general overview of the implementation using Eclipse. Section 4 explains the developed tool for managing traceability. Section 5 gives a cost model of the approach. Section 6 details an evaluation of the approach based on a survey and a comparison of the developed tool with two HA commercial tools. Finally, Section 7 is dedicated to conclusions and future work.

2. Foundations and related work

2.1. Home automation systems development

At the present time, developers of HA applications mainly use software tools provided either by the device manufacturer, in the case of proprietary system, or by the associations responsible for providing support for the technology in the case of the standard systems. These tools are usually platform-dependent, code generation-oriented integrated environments which do little to raise the level of abstraction. Moreover, the concrete syntax that they use is not usually very intuitive, so that the user requires very specialized training and can only work in the immediate context of the solution.

The whole process of development of HA applications is carried out by an expert in the domain who collates the customer's requirements for an installation (elements to be integrated, services required, selection of a concrete technology, etc.) based on his own experience. This expert carries out the selection and deployment of the devices and afterwards programs them (using a platform-dependent development infrastructure) so as to achieve the desired functionality. Working in this manner it is rather difficult to achieve some of the desired attributes of software systems such as interoperability, flexibility, re-use and productivity. Besides, tools to develop projects are completely different in each platform, so learning a new technology implies new training. Thus, developers usually focus on a particular technology, leaving aside other platforms. This is due to the long training time and high specialization required (a good developer would need to have undertaken around 100 h of training and have months of practice).

2.2. Related work

2.2.1. MDD for HA development

The literature offers a few examples of works which try to reach in an integrated way the development of HA systems using an MDD approach. Among these it is important to highlight the works of Muñoz et al. (2006), Voelter and Groher (2007) and Nain et al. (2008) that outline the necessity of using a model driven approach in HA systems development. The aim is to increase the level of abstraction, the productivity and the quality of the software, besides maintaining the independence of the implementation platform. These proposals represent a good example of the advantages that the use of MDD offers in the development of HA systems, but they also present some drawbacks. In the first place, Muñoz uses the UML notation for requirement elicitation which is not very intuitive for experts in the field of HA. In the work of Voelter, a set of HA devices is defined in the meta-model. Applications are created using the tool named Tree Editor provided by the plug-in EMF for Eclipse. Hence to use an HA device not included in the meta-model, it is necessary to build a new meta-model or extend the existing one. Nain presents EnTiMid as a middleware composed of several layers. A driver layer is in charge of the connection between the devices and the Unified Service layer. A bridge layer links the Unified Service instances to diverse service technologies such as Universal Plug And Play (UPnP) and Devices Profile for Web Services (DPWS) (Jammes et al., 2005). The work defines EnTiMid as a middleware implementation that supports various services access models and also describes how these artifacts are generated using MDD.

In these proposals the code generation is oriented to obtain OSGi (Open Service Gateway Initiative) drivers for a server or middleware platform, and not to the programming of the HA devices. Therefore, it will always be necessary an expert of the specific platform to program these devices.

Contrary to the previous examples, in our framework the level of abstraction and usability of requirements modeling rises with the use of a DSL that uses specific concepts of the domain. In addition, our proposal guides the code generation to the automatic programming of the devices of the selected HA technology. In this way the need for specific knowledge of each platform is avoided, as well as the intervention of an expert in the technology.

2.2.2. Reuse of DSL models

The literature distinguishes between two general types of reuse approaches (von Knethen et al., 2002): composition and generation-based approaches. Composition-based approaches are based on composing reusable assets. This type of approach is typically applied for design or code reuse. Generation-based approaches focus on instantiating reusable abstractions. Popular