Journal of Network and Computer Applications 35 (2012) 1086–1105

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

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

A model-driven development method for collaborative modeling tools

Jesús Gallardo[a,*], Crescencio Bravo[b,1], Miguel A. Redondo[b,1]

[a] Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Escuela Universitaria Politécnica de Teruel, Ciudad Escolar s/n, 44003 Teruel, Spain
[b] Departamento de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, Escuela Superior de Informática, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 12 May 2011
Received in revised form
27 October 2011
Accepted 5 December 2011
Available online 22 December 2011

Keywords:
Collaborative modeling
Network modeling environments
Model-driven development
Groupware
Computer-supported collaborative work

ABSTRACT

Collaborative modeling tools are useful for many tasks within design or learning processes. However, they are difficult to build and are usually domain-specific. In response to this situation, we propose a model-driven method for the development of domain-independent collaborative modeling tools. This method consists of a methodological framework, a conceptual framework and a technological framework. The methodological framework defines the phases to be carried out when applying the method, whilst the conceptual framework is made up of the meta-models used in the method and the transformation processes established between them. Finally, the technological framework consists of the integration of some plug-ins from the Eclipse Modeling Project with some add-ons which provide collaborative functionality. Some case studies that exemplify this development method for specific domains are explained in detail, together with comparisons with similar approaches. Thus, an initial evaluation of this approach is provided and some advantages over those other approaches are identified. A further evaluation process in the form of an empirical study of use has also been carried out. Hence, the method proves to be useful for any user who does not have advanced knowledge of groupware programming and who needs to use a collaborative modeling tool in his/her work. Moreover, each framework implies a contribution that can be used in different contexts.

1. Introduction

Today's society has an increasing need to communicate and collaborate in the many different tasks carried out in the workplace or in learning or leisure spaces. This requires a new type of software designed for groups, which is known as groupware. The design of groupware tools is an emerging field of Software Engineering, and is gaining in importance (Gerosa et al., 2005). Aspects such as network management, synchronization, awareness and shared workspaces are new concepts which have been introduced with groupware. Unfortunately, groupware is more difficult to design and evaluate than a non-collaborative application since, among other reasons, social protocols and group activities must be taken into account (Grudin, 1993). As such, groupware development needs specific techniques that take into account the aforementioned particularities of this type of software.

In this study, we tackle the problem of building collaborative tools to support modeling tasks. We focus on a specific type of groupware; that of domain-independent modeling tools. In these tools, several distributed designers typically interact to construct a model or artifact, working in shared workspaces which are based on the metaphor of a whiteboard. The model is built according to a specified goal or task. What is unusual about the kind of tools we are dealing with, i.e. domain-independent tools, when compared to other modeling systems covered in the literature (see Section 2), is that the design (model) to be created is not restricted to a specific domain; that is to say, the tool is able to deal with diverse design scopes described by means of a configuration process (Greenfield, 2005). This is in sharp contrast to domain-specific tools, which allow for the building of diagrams within a specific domain, such as those of digital circuits, concept maps, any kind of UML diagrams, etc. Furthermore, these tools could be made up of different workspaces in order to carry out tasks other than the modeling task itself. An example of this could be a task for arranging the work that is to be done among the different users, or a task for doing some sketches prior to the modeling task.

Thus, the research hypothesis we pose here states that it is feasible to provide systematic support for the development of such systems. This is addressed in this study with the proposal of a full model-driven engineering method for the construction of domain-independent modeling groupware tools. Within this method, the use of software engineering techniques such as meta-modeling is very useful. With meta-modeling, we define the structure of the models that are to be built using the modeling tool, as well as the structure of the tool itself, so that meta-models

* Corresponding author. Tel.: +34 978 61 81 02.
E-mail addresses: jesus.gallardo@unizar.es (J. Gallardo), Crescencio.Bravo@uclm.es (C. Bravo), MiguelRedondo@uclm.es (M.A. Redondo).
[1] Tel.: +34 926 29 53 00; fax: +34 926 29 53 54.

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.12.009

become the basis for building the models that characterize a specific tool (Gallardo et al., 2007). Our method, therefore, makes it possible to work with the same modeling tool in different domains. Take for example a software development company which wants its employees to carry out a discussion using both a flow chart and a class diagram. With our method, this could be done using the same tool, which will be adapted on each occasion to each specific domain.

The development method is intended for users with no advanced knowledge of software programming; the process for defining new application domains is simple and descriptive, and the generation of the final groupware tool is guided by the tools supporting the method. Nevertheless, users can be helped by experts in the application domain, or software engineers if necessary.

Our model-driven development method is based on three frameworks: a methodological framework, a conceptual framework and a technological framework. The methodological framework consists of a series of phases that must be followed by the non-expert user who wishes to develop a collaborative modeling tool. These phases are: (i) the identification of the domain, (ii) the modeling of the domain and the workspaces, (iii) the production of the collaborative modeling tool, which includes the model transformations and the generation of the tool itself, and (iv) the use of the generated tool. The conceptual framework is made up of the models that are used in the meta-modeling process. These models are mainly the domain and workspace meta-models. And, finally, the technological framework consists of a series of plug-ins for the Eclipse platform that have been modified and extended to generate collaborative applications. With these frameworks, comprehensive support for the entire development method is provided.

Our first attempt at building a domain-independent collaborative tool was the SPACE-DESIGN tool (Gallardo et al., 2011b). SPACE-DESIGN is domain-independent, since the tool reads the domain specification from an XML file and spawns the corresponding user interface to carry out the modeling process. This tool includes some fixed awareness and coordination mechanisms implemented as reusable components: tele-pointers, a session panel, a floor control tool, an interactions list and a structured chat tool. In the field of groupware, awareness is defined as the perception of the group and its activity (Dourish and Bellotti, 1992). The present work is an evolution of that done with the SPACE-DESIGN tool and improves certain elements, such as the way in which the domain is specified, the extension of the working process to cover new workspaces, and the reusability and extensibility of the generated tool. Some other ways of carrying out modeling tasks in a collaborative way have been considered (sharing a non-collaborative modeling tool, for example) but they were all ruled out for one reason or another, as explained later.

This article continues by presenting the foundations of collaborative modeling tools and, in particular, of domain-independent tools. In the third section, we explain the concepts and models identified in the meta-modeling process. The fourth section describes in detail our development method for collaborative modeling tools. Then, we discuss the application of our proposal in some case studies. Section 6 is about the empirical study of use carried out to test the suitability of our approach. Finally, in Section 7, we draw some conclusions and outline future work.

2. Related work

2.1. Domain independence in groupware tools

Of the various types of groupware tools, this paper focuses on distributed synchronous tools to support the construction of a model made up, at a conceptual level, of a set of objects and the relationships between them. Designers work in a shared workspace, where they create a model collaboratively. To achieve this, users participate in design sessions in which they make use of domain objects, relationships and other graphical elements found on the workspace toolbars. Typically, the final model is built in response to a specific goal. The setting may be a group work activity, where the problem is a real situation to be dealt with in the scope of a company or institution or an e-learning system, where a learning method based on collaborative problem solving is followed (Bravo et al., 2006). As mentioned above, the whole system can be made up of other workspaces apart from the modeling one in order to give support to other steps of the design process.

The second basic characteristic of the modeling tools we are looking at is that they are independent of the domain that is being considered. At this point, we can define a domain as the particular syntax and semantics that are used to construct models. In such tools, therefore, the domain is not fixed but instead can be defined by the user employing suitable authoring tools.

These domain-independent modeling tools present some very typical problems. These include the materialization of the shared workspaces, floor control policies, coordination and communication processes, and the definition of the domains. In the following paragraphs, we discuss some examples of tools related to this study, some of which are domain-independent and others which are domain-dependent.

Cool Modes (Pinkwart et al., 2002) is a cooperative modeling tool in which several users work together in order to design a model or artifact. It includes a set of plug-ins containing the elements that can be placed on the whiteboard. The strong point of the tool is that the constructed model can be simulated, since the plug-ins provide this functionality. However, the definition of plug-ins is not very versatile, since they are programmed within the code itself.

A similar tool is Synergo (Avouris et al., 2004), which has several fixed palettes containing design components that can be connected to each other. Designs are stored in a proprietary format, whilst the different modeling actions are stored in an XML file which can be analyzed later. Another feature of Synergo is its communication tool (chat feature), which allows users to have discussions amongst themselves.

There have been a few attempts at developing collaborative modeling tools using the plug-ins included in the Eclipse Modeling Framework (EMF), which is the technological approach we have followed. One of those attempts is the Dawn project (Flügge, 2010). However, the tools generated in this project do not have any awareness or collaboration support elements, and they are only made up of the modeling workspace. These are the main differences between Dawn and our work, which gives great importance to awareness and collaboration support.

In contrast to the described tools, we can also mention some examples of domain-dependent modeling tools, which are conceived and designed to work in a specific application domain. DomoSim-TPC (Bravo et al., 2006) is one such tool, which works in the Domotics domain. Concretely, DomoSim-TPC is a modeling tool which has a shared workspace in which the users synchronously build a model as a solution to a previously defined problem. DomoSim-TPC includes a number of awareness and discussion elements: a session panel, an interactions list, a structured chat tool and a decision-making tool. Another final example is Co-Lab (van Joolingen et al., 2005). Co-Lab is an environment designed for synchronous collaborative inquiry learning that works within the System Dynamics domain. An important difference between Co-Lab and other modeling tools is that Co-Lab uses the metaphor of a building, wherein each building represents a course in a specific domain. Co-Lab also features a number of awareness mechanisms,