proceedings

MDPI

Proceedings

Collaborative Modeling of Group Learning Applications Using Eclipse Technology †

Yoel Arroyo [1,*], Ana I. Molina [1], Miguel A. Redondo [1], Jesús Gallardo [2] and Carmen Lacave [1]

[1] Escuela Superior de Informática, Universidad de Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain; AnaIsabel.Molina@uclm.es (A.I.M.); Miguel.Redondo@uclm.es (M.A.R.); Carmen.Lacave@uclm.es (C.L.)

[2] Escuela Universitaria Politécnica de Teruel, Universidad de Zaragoza, Calle Atarazana, 2, 44003 Teruel, Spain; Jesus.Gallardo@unizar.es

* Correspondence: Yoel.Arroyo@uclm.es; Tel.: +34-926295300-96676

† Presented at the 13th International Conference on Ubiquitous Computing and Ambient Intelligence UCAmI 2019, Toledo, Spain, 2–5 December 2019.

Published: 20 November 2019

Abstract: The design and creation of groupware tools is a complex task that usually requires the participation of different stakeholders (software engineers, designers, etc.), either working at the same time or collaborating asynchronously. This paper describes an innovative model-driven development process to support the collaborative modeling of group learning applications, as well as the Computer Aided Software Engineering (CASE) tool that technologically supports it, the Learning Collaborative Interactive Applications Tool (Learn-CIAT) graphical editor. In its development, we applied technologies integrated within the Eclipse platform. The processes and tools described in this paper supply an important contribution to systematize the design and development of these kinds of applications.

Keywords: Computer-Supported Cooperative Work (CSCW); Computer-Supported Collaborative Learning (CSCL); Model-Driven Engineering (MDE); Technology Enhanced Learning (TEL) systems; e-learning

1. Introduction

In the last few years, substantial production of software systems for group learning was observed, at both the research and application levels, reaching very remarkable usage levels. This was achieved in part thanks to the use of learning content management systems (LCMSs) or massive online open courses (MOOCs) [1]. The design and creation of groupware tools is, therefore, an emergent field in software engineering that gained relevance and interest recently, demanding new systematic procedures for its development. Nevertheless, when developing these kinds of tools, there are aspects such as synchronization, coordination, or the existence of shared workspaces which did not exist in single-user systems. In addition, the complexity of providing these systems with user interfaces that are usable and that promote group awareness must be considered, as they are a fundamental requirement in the case of synchronous collaborative systems [2].

Complex design tasks as the one described usually require the participation of different stakeholders (software engineers, designers, teachers, etc.), either working at the same time or collaborating asynchronously. Therefore, we consider it necessary to have a collaborative design environment that allows the members of a work team to share information and coordinate their modeling activities. With this purpose, we present in this paper a new model-driven engineering approach to support the collaborative modeling of group learning applications, as well as the CASE tool that technologically supports it, the Learn-CIAT graphical editor. MDE has the potential to

greatly improve current practices in software development such as increasing the productivity of the developer and the maintainability of the developments, capturing domain knowledge, improving communication among stakeholders, considering models as long-term assets, or making the delay of technological decisions possible, among other benefits.

This paper is structured as follows: Section 2 introduces some previous studies related to this work. Section 3 presents the set of technologies and languages within the Eclipse platform solutions that are necessary to create graphical editors like Learn-CIAT. Section 4 describes how the learning applications collaborative design process is supported by using Learn-CIAT. Finally, some conclusions and remarks derived from the work realized are enumerated.

2. Previous Works

The development of groupware applications for the educational field is a particularly challenging area which is addressed under the paradigm called computer-supported collaborative learning (CSCL) [3]. This paradigm is at the intersection of several knowledge areas: educational practice, learning psychology, and the support that information and communication technologies (ICT) provides to the teaching/learning processes. Therefore, to the already complex task of developing a usable collaborative system, the peculiarity of considering a large number of aspects related to psych-pedagogy must be added, such as formation and configuration of groups of students, alignment of the tasks to be performed with the learning objectives, quality and cognitive load imposed by the didactic materials to be used, consideration of the student's prior knowledge, their learning style or motivation, adequate support for problem solving, discussion, argumentation, and decision-making, etc. As a solution to this problem, the authors of this paper proposed the Learn-Collaborative Interactive Applications Notation (Learn-CIAN) notation [4] (Figure 1).

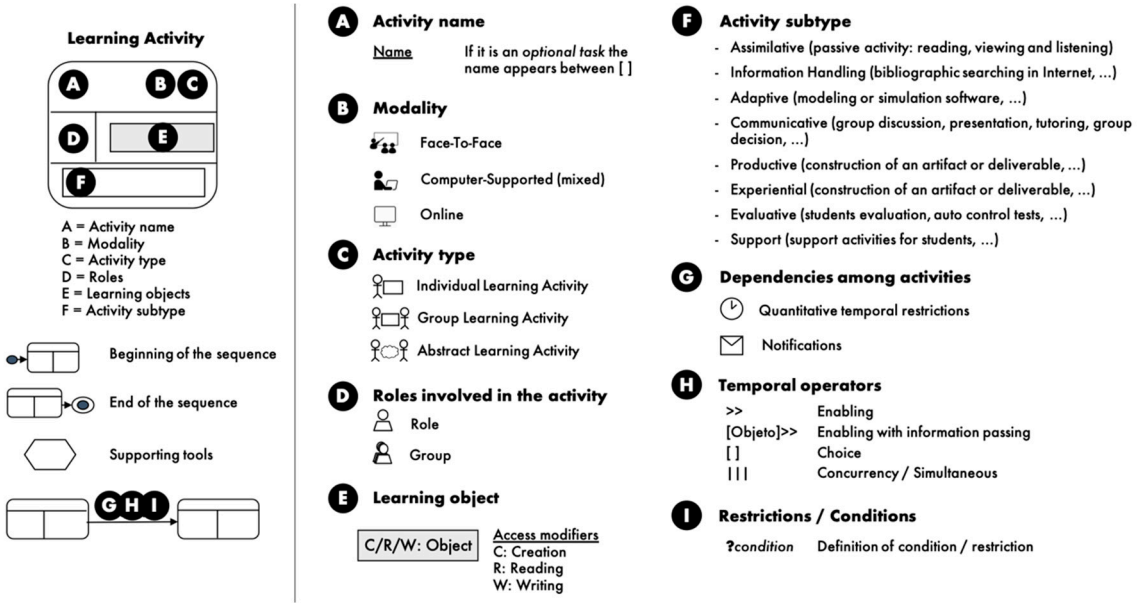

Figure 1. Learn-CIAN notation.

This notation is prepared to support the design of flows of learning activities within the framework of CSCL systems. Nonetheless, it still does not include aspects related to *awareness* (characteristic in the design of synchronous collaborative systems) [5], nor aspects related to the so-called *pedagogical usability* (characteristic of learning systems) [6]. According to Gutwin and Greenberg [2], awareness is defined as the knowledge and perception of the working group and its activity. Its main objective is, therefore, to reduce the effort needed to carry out group activities. Among the works that apply the MDE principles and contemplate aspects of awareness, the SpacEclipse [7] development method stands out. SpacEclipse is technologically supported in the form of an Eclipse plug-in. This makes it possible to generate, through a semi-automatic model-driven

process, synchronous collaborative modeling systems adapted to any domain or type of diagram. To this end, it incorporates a series of conceptual frameworks that contain, among other things, a fairly broad set of awareness components (telepointers, radar views, session panels, etc.).

The integration of the Learn-CIAN notation with an updated version of SpacEclipse (which contemplates more tasks than just modeling) would allow the automatic generation of CSCL systems with adequate awareness support and full functionality. Nevertheless, the aspects most related to learning must also be covered, i.e., the pedagogical usability of the systems to be generated. The purpose of pedagogical usability is not to qualify a didactic material as "good" or "bad", but to help users to choose the most appropriate alternative for each concrete learning situation [8]. Therefore, the Learn-CIAN notation should also include a series of learning components or heuristics that could be reflected in the generated CSCL systems. In this regard, the mobile learning evaluation framework (MoLEF) [9] stands out among the existing works that help to achieve this objective. MoLEF is a framework for the evaluation of mobile learning (m-learning) applications, including an evaluation instrument (a questionnaire) called the M-Learning Applications Quality Evaluation Questionnaire (CECAM). This questionnaire consists of a total of 56 items, 29 of which refer to pedagogical usability. These items can be used as heuristics to guide the design of learning systems, or as an evaluation checklist. In addition, CECAM was refined to analyze its validity and reliability [10], so that it can be considered a fairly reliable instrument.

The combination of these works made it possible to formalize a new methodological proposal called Learning Collaborative Interactive Applications Methodology (Learn-CIAM)—a systematic method which guides software engineers in the design of group learning applications through a model-based development process. This proposal is technologically supported by a collaborative tool, the Learn-CIAT graphical editor, which is presented in Section 4 of this article. Firstly, we present the set of technologies and languages required for the creation of these kinds of tools and the model-driven development process proposed.

3. Eclipse Technologies and Languages

Between all existing works that allow to apply a model-based approach on Eclipse, it is worth highlighting the Extensible Platform of Integrated Languages for Model Management (Epsilon) project (https://www.eclipse.org/epsilon/) [11]. Epsilon is a family of task-specific programming languages, consistent and interoperable. It arose with the idea of facilitating even more the common modeling tasks on Eclipse such as the generation of code, the transformation between models, or the validation of models, among others.

In view of the specific objectives of this research work, we would like to highlight only the languages required for the creation of the Learn-CIAT graphical editor: the Epsilon object language (EOL), the Epsilon validation language (EVL), and the Epsilon generation language (EGL). EOL, considered to be the core of Epsilon, is an imperative language for creating, consulting, and modifying models. Its syntax is very similar to that of Java, which is a great advantage for developers accustomed to working with object-oriented languages. EVL, on the other hand, is an EOL extension that allows developers to define restrictions (similar to the Object Constraint Language (OCL)), customize error messages received by the user, and define quick-fixes to validation problems. Meanwhile, EGL is an extension of EOL, facilitating the generation of code of the final systems from a specific model. It is a markup language based on templates with a syntax similar to that of PHP.

In addition to a set of high-level languages, Epsilon also incorporates a set of tools and utilities that complement interaction such as a graphical editor generation assistant (Eugenia), a Human-Usable Textual Notation (HUTN) implementation, a customizable tree-like editor (Exceed), and a multi-view editor to establish cross-references between models (Modelink). Of all of them, Eugenia is the most interesting for the resolution of the objectives of this work.

Eugenia [12] is a tool capable of automatically generating the models needed to create a graphical editor from only one annotated Ecore meta-model, written in Emfatic (https://www.eclipse.org/emfatic/) language (Figure 2). The Emfatic language allows to represent Ecore meta-models textually with a syntax very similar to that of Java. Eugenia's main goal is to

reduce the complexity of creating Eclipse modeling framework (EMF) and graphical modeling framework (GMF) models, which forces developers to implement and maintain proprietary models manually, making the job considerably more difficult.

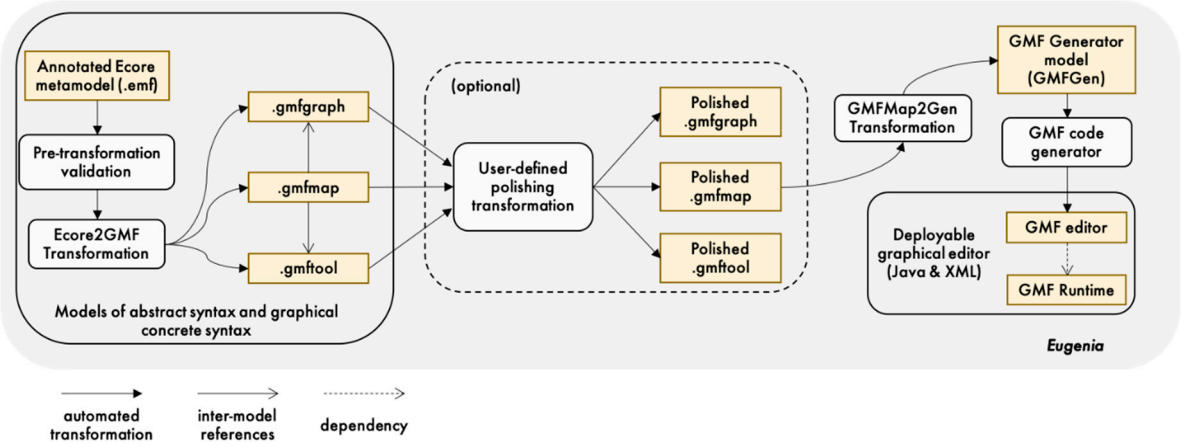

Figure 2. Outline of the Eugenia development process.

To this end, Eugenia incorporates a set of annotations that can be added to the main Ecore metamodel. This way, Eugenia is able to execute a series of automatic transformations until generating the intermediate models and the final graphical editor, thus increasing the GMF level of abstraction. Nevertheless, the set of annotations it incorporates, around six different annotation categories, may become insufficient as the complexity or requirements of the desired graphical editor increases. To address this drawback, Eugenia has a high capacity for customization over the generated graphical editor. This is achieved through the definition of three additional independent files, written in EOL language: *ECore2GMF.eol*, *FixGenModel.eol*, and *FixGMFGen.eol*. Eugenia allows developers to automatically incorporate the customizations implemented on these files just by starting the automatic generation process of the graphical editor. While the first allows consulting, modifying, and deleting the information of the editor's own models, the second and third allow configuring the dependencies of the final plug-in, allowing the user to add external functionality and attach it to the graphical editor (e.g., it is usual to add an additional plug-in containing a battery of images to attach them to the nodes of the graphical editor).

The main advantage over the EMF/GMF-based process is that, thanks to these files/templates, it is no longer necessary to manually re-code these changes each time developers want to make a new modification on the graphical editor already generated. This way, each time Eugenia starts a new generation process, the final graphical editor applies these changes automatically. Thus, Eugenia not only reduces the number of models to be defined by the developer, but also encourages their reuse in different work contexts.

A Model-Driven Development Proposal for the Generation of Graphical Editors

In Figure 3 it is shown schematically how the process for obtaining the Learn-CIAT tool is organized, as well as the technologies and specification languages used in its development. The following are the phases in the creation of a tool of this kind:

1. Informal specification. At this stage, the developers must generate some first "sketches" of the graphical editor, specifying the working domain. As previously mentioned, using Eugenia, graphical editors with different domains could be elaborated in a simpler way, for example, graphical editors for the modeling of digital circuits, network topologies, or family trees. Nevertheless, in the context of this research work, we want to generate Learn-CIAT, the graphical editor that supports Learn-CIAM for collaborative modeling of group learning applications.

2. Domain modeling. During this stage, developers must proceed to model the previously defined domain. To this end, the developer might be an expert of the technology used, identifying and relating each of the parts of the graphical editor "sketch" in the definition and elaboration of the main meta-model. Specifically, the developer must generate the Ecore representation (identifying all the corresponding EPackages, EClasses, EReferences, EAtributtes). Due to the fact that the technology used is Eugenia, it is necessary to add to this representation the annotations that it incorporates and which facilitate the automatic generation of the graphical editor. All this information (meta-model + annotations) must be implemented in an Emfatic file.

3. Graphical editor customization. By using Eugenia, users will be able to generate graphical editors by implementing one Emfatic file. Nevertheless, when the desired graphical editor is more complex than Eugenia technology can provide/generate using its basic annotations, it will be necessary to modify the figures and relationships it generates by default to adapt them to the graphical editor needs. To this end, it is necessary to generate (or reuse) an EOL template that contains these modifications. This template will be detected and applied automatically by Eugenia in the graphical editor generation process. Also, sometimes, the restrictions that are generated automatically from the meta-model will not be enough, and it will be necessary to add more manually. To address this problem, one or several templates in EVL language must be implemented, incorporating the restrictions that the graphical editor must additionally support. According to Figure 3, these two problems correspond to the first cubicle of the step.

In addition, the developers have the chance to provide collaborative functionality to the graphical editor. An updated version of the SpacEclipse plug-in was prepared for this purpose. This plug-in contains the basic code necessary to receive single-user graphical editors and provide them with collaborative functionality, thus generating a collaborative graphical editor automatically for Eclipse. This is possible by implementing or reusing an EGL-based patch which, when applied automatically, adds the necessary collaborative functionality to the graphical editor. The same way, another interesting option provided by the technology used is that of being able to modify the code of the graphical editor generated by Eugenia using templates coded in EGL. This is usually necessary in those cases in which, despite having made some visual modifications following the first stage of this step, the generated graphical editor still does not meet expectations. For example, it may be necessary to bring the nodes closer to the label of one of their attributes.

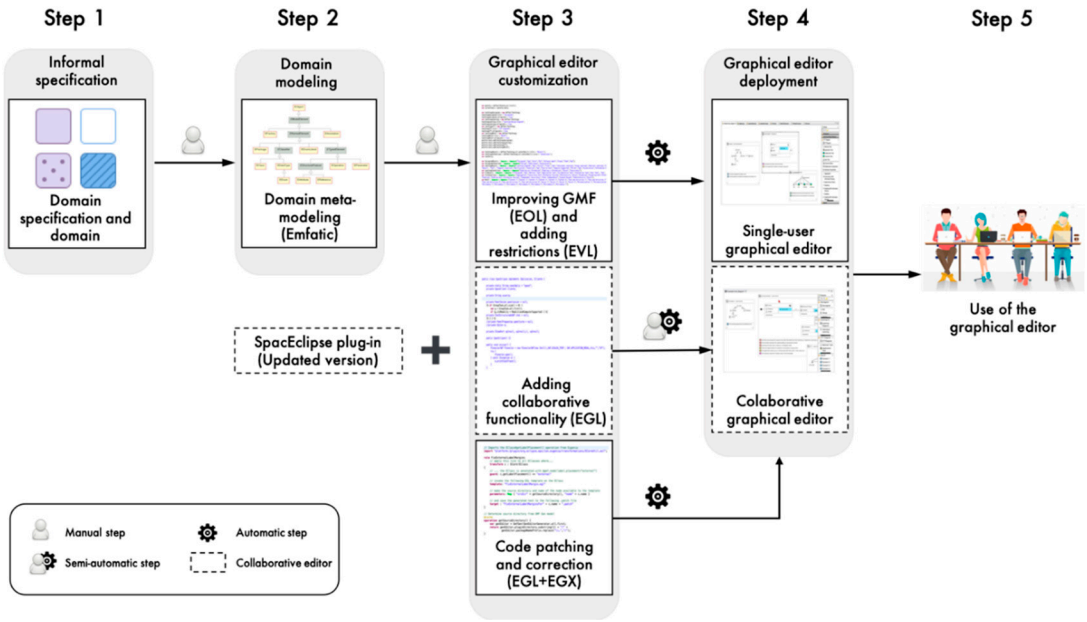

Figure 3. Outline of the model-driven development process.

4. Graphical editor deployment. Once the graphical editor is generated and it was verified that its functionality is the desired one, it is necessary to proceed to the deployment of a stable version. Basically, it is necessary to generate an installer which includes the functionality that assures the correct functioning of the graphical editor in other instances of Eclipse. In order to share it with the end users, different ways of installation can be provided, either physically (USB, HDD, etc.) or via the web (through Eclipse internal shop or an external URL).
5. Use of the graphical editor. Finally, when the end users install the plug-in on their Eclipse platform, they will be able to start making designs on the graphical editor.

4. Collaborative Modeling of Group Learning Applications

As any Eclipse-based graphical editor, Learn-CIAT is composed of a canvas (drawing area) and a palette containing the nodes and relationships which can be dragged and instantiated over it in the design of each new diagram. Thus, using Learn-CIAT, it is possible to model and configure both learning courses and applications using the next components: the Learn-CIAN notation for modeling the flows of learning activities (Figure 4a), the workspace and awareness configuration of group learning applications (Figure 4b), and the criteria related to pedagogical usability (Figure 4c). Additionally, it incorporates the definition of sociograms [13], a diagram that allows to establish the hierarchy of the roles involved in a given learning course. At the same time, since SpacEclipse was used in its elaboration, Learn-CIAT incorporates a series of components/widgets to facilitate the collaborative modeling work to the final users. Attending Figure 4, the workspace of the collaborative tool is composed of (1) a session panel, which allows to see the set of users that are co-working in the design; (2) a project explorer; (3) a chat; (4) a properties view; (5) a turn-taking panel, necessary to control the synchronous collaborative modeling process; (6) a radar view, which is really useful when the size of diagrams becomes bigger; and (7) a problems view, required to validate the diagrams looking for errors/warnings. Moreover, each of these components has its own awareness characteristics. For example, a session panel attaches a color for each user, even showing the exact user who is modeling at the moment (label editing…). The chat incorporates a structured module with a set of typical phrases in graphical modeling processes. The turn-taking panel incorporates a semaphore and sounds to indicate a shift change request (green when there is one and red when not) and a voting system to accept it or deny it. All these components are in two languages (Spanish and English), facilitating the work and trying to reach a greater number of users.

Another important feature of the Learn-CIAT graphical editor is the inclusion of runtime validations. Because the graphical editor was generated by applying a model-driven development process, some restrictions are already imposed implicitly by the meta-model. Nonetheless, for those restrictions that the technology would not be able to automatically incorporate, a set of additional EVL rules was defined. Again, attending Figure 4, the workflow of a Java programming course and the configuration of a collaborative code editor is presented. The graphical editor detected up to three errors and a warning after the model validation process, whereby the user needs to select the position of the console view in the user interface workspace; an error is shown indicating that a code editor must contain a turn-taking panel; a computer-supported group activity named “practical project” is declared as adaptive, when these types of tasks are productive (according to the rules established in Learn-CIAN and the configuration of workspaces) [4]; finally, to show the flexibility that EVL provides, in the sociogram (bottom of the figure) appears a warning indicating that the name of the professor role should start with capital letters. The user has the possibility of manually correcting each of these errors or warnings, or start its corresponding quick fix, which, if it was declared in the EVL template, is able to start an automatic process that solves it only with the push of a button.

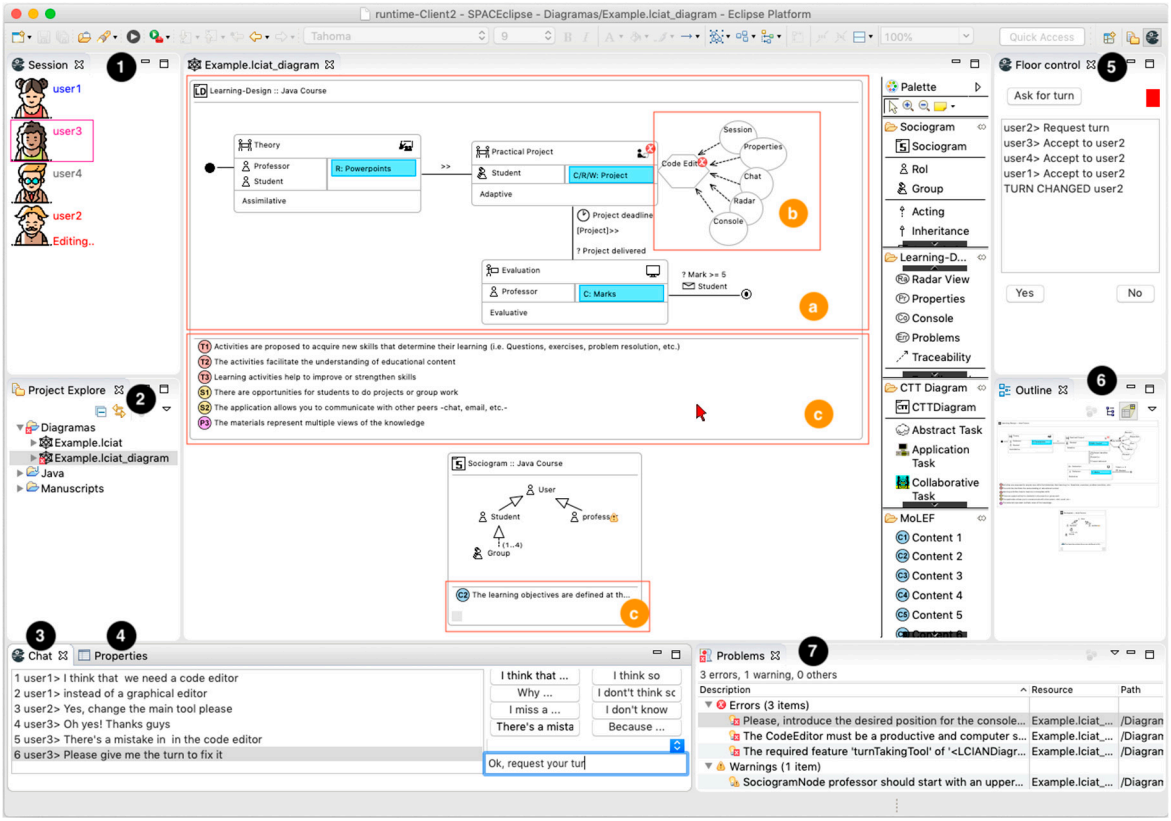

Figure 4. A collaborative modeling and validation process on Learn-CIAT.

5. Concluding Remarks

The implementation of applications to support group learning is a complex process that involves considering different perspectives (collaboration, interaction, awareness, pedagogical, etc.) and stakeholders (teachers, software engineers, etc.). In this paper, an innovative model-driven development approach to support the collaborative modeling of these kinds of applications was described at a methodological and technological level. This development paradigm was chosen since it guarantees a series of advantages compared to others such as traditional ad hoc. The use of conceptual frameworks as input devices implies a faster and economic development process, as well as favoring their reuse in the resolution of other problems, regardless of the specific scope of work for which they were originally intended.

The proposed framework takes the Learn-CIAM methodological proposal as a starting point. In this framework, aspects required for the design and creation of group learning systems such as awareness and pedagogical usability are included. Its collaborative modeling is supported by the Learn-CIAT tool, in which we applied innovative technologies integrated within the Eclipse platform during development, an environment widely used in both research and industrial contexts.

The processes and tools described in this paper make an important contribution to systematize the design and development of applications supporting CSCL. Its use should facilitate the work of software engineers (or groupware engineers) and teachers, who are responsible for designing and developing these kinds of applications, which are currently used en masse.

Acknowledgments: This work was partially funded by the project TIN2015-66731-C2-2-R of the Ministry of Science, Innovation, and Universities.

References

1. Rius, À.; Conesa, J.; García-Barriocana, E.; Sicilia, M.Á. An ontology-driven framework for specifying, adapting and implementing educational settings. *Appl. Ontol.* **2017**, *12*, 33–58. doi:10.3233/AO-170176.

2. Gutwin, C.; Greenberg, S. Workspace awareness for groupware. In Proceedings of the Conference companion on Human Factors in Computing Systems Common Ground—CHI'96, Vancouver, BC, Canada, 13–18 April 1996; pp. 208–209. doi:10.1145/257089.257284.
3. Koschmann, T. *CSCL: Theory and Practice of an Emerging Paradigm*; L. Erlbaum Associates. 1996. Available online: https://psycnet.apa.org/record/1997-97125-000 (accessed on 20 July 2018).
4. Molina, A.I.; Arroyo, Y.; Lacave, C.; Redondo, M.A. Learn-CIAN: A visual language for the modelling of group learning processes. *Br. J. Educ. Technol.* **2018**, *49*, 1096–1112. doi:10.1111/bjet.12680.
5. Collazos, C.A.; Gutiérrez, F.L.; Gallardo, J.; Ortega, M.; Fardoun, H.M.; Molina, A.I. Descriptive theory of awareness for groupware development. *J. Ambient Intell. Humaniz. Comput.* **2018**, doi:10.1007/s12652-018-1165-9.
6. Lee, Y.Y.; Lowe, M.S. Building Positive Learning Experiences through Pedagogical Research Guide Design. *J. Web Librariansh.* **2018**, *12*, 205–231. doi:10.1080/19322909.2018.1499453.
7. Gallardo, J.; Bravo, C.; Redondo, M.A. A model-driven development method for collaborative modeling tools. *J. Netw. Comput. Appl.* **2012**, *35*, 1086–1105. doi:10.1016/j.jnca.2011.12.009.
8. Nokelainen, P. An empirical assessment of pedagogical usability criteria for digital learning material with elementary school students. *Educ. Technol. Soc.* **2006**, *9*, 178–197. Available online: https://www.j-ets.net/ETS/journals/9_2/15.pdf (accessed on 14 February 2019).
9. Navarro, C.X.; Molina, A.I.; Redondo, M.A.; Juárez-Ramírez, R. Framework to Evaluate M-Learning Systems: A Technological and Pedagogical Approach. *Rev. Iberoam. Tecnol. del Aprendiz.* **2016**, *11*, 33–40. doi:10.1109/RITA.2016.2518459.
10. Arroyo, Y.; Molina, A.I.; Lacave, C.; Redondo, M.A.; Ortega, M. The GreedEx experience: Evolution of different versions for the learning of greedy algorithms. *Comput. Appl. Eng. Educ.* **2018**, *26*, 1306–1317. doi:10.1002/cae.22023.
11. Kolovos, D.S.; Paige, R.; Rose, L.; Polack, F. *The Epsilon Book*. 2010. Available online: https://www.eclipse.org/epsilon/doc/book/ (accessed on 1 November 2019).
12. Kolovos, D.S.; García-Domínguez, A.; Rose, L.M.; Paige, R.F. Eugenia: Towards disciplined and automated development of GMF-based graphical model editors. *Softw. Syst. Model.* **2017**, *16*, 229–255. doi:10.1007/s10270-015-0455-3.
13. Molina, A.I.; Redondo, M.A.; Ortega, M.; Hoppe, U. CIAM: A methodology for the development of groupware user interfaces. *J. Univers. Comput. Sci.* **2008**, *14*, 1435–1446. doi:10.3217/jucs-014-09-1435.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).