US2StarTool: Generating i* Models from User Stories

Renato Mesquita[1], Aline Jaqueira[1], Celso Agra[2], Márcia Lucena[1], and Fernanda Alencar[23]

[1]Departamento de Informática e Matemática Aplicada – UFRN
{rmsnatal,alineopj}@gmail.com; marciaj@dimap.ufrn.br
[2]Programa de Pós-Graduação em Engenharia da Computação – UPE
[3]Departamento de Eletrônica e Sistemas - UFPE
celsoagra@gmail.com; fernanda.ralencar@ufpe.br

Abstract. In agile methods, the requirements are represented by user stories. However, this model does not allow a good visualization of context in which a story is inserted, reducing the understanding of the system as a whole. On the other hand, the i* model presents dependencies among organizational actors, and the understanding of the context in which a requirement is inserted. This paper presents an implementation of an automated solution as a tool to mapping user stories into i* models, US2StarTool, adding a support for agile development environment. US2StarTool can help requirements engineer in agile development environments contextualizing the environment in which user stories are inserted and showing relationships between the actors and the system.

Keywords: User Story, i* Model, Meta-model, Model Mapping.

1 Introduction

To build a software project, it must be specified the requirements to be satisfied. The requirements elicitation aims to specify the software completeness and correctness, besides guarantee the quality, validation and acceptance. In agile methods, the requirements are developed of incremental way, according to stakeholders demands. The artifacts used are user stories. To represent a user story, Cohn [2] suggests a common format: As <role>, I want <action>, to <goal>. However, in user stories you cannot visualize dependencies among stories [3], besides it is hard to assimilate the context that they are included within a system [8].

The i* modeling technique [9] is one of the most relevant Goal-Oriented Requirements Engineering (GORE) approaches and provides a view of involved actors and their dependencies. Thus, the requirement model described from i* models provides a complete representation of requirements. The dependencies between actors are represented and by this model it is possible understand the context.

The benefits of using visual models to describe the requirements are presented in [1]. Thus, this paper presents the needed stages to develop a tool named US2StarTool, able to map user stories into i* model based on a set of mapping heuristics. In the Section 2 we define the research objectives. Then Section 3 explains how the tool has been developed. In Section 4 we discuss the scientific contributions. Section 5 provides the conclusions and Section 6 presents future works.

2 Objectives of the research

This paper presents an automated solution, named US2StarTool, that the main objective is mapping user stories into i* models. The results are used as a complementary tool for agile development process, providing a better understanding for context in the projects based on user stories, dependencies between actors involved and the whole system-to-be.

3 Development and required tools

To implement the US2StarTool, we used the EuGENia tool, a small part of the Epsilon Framework (Extensible Platform for Specification of Integrated Languages for Model Management). We consider that EuGENia tool is used as code generation, models transformation, validation, comparison and refactoring with EMF (Eclipse Modeling Framework) and other types of models. The EMF is a modeling framework and code generation to build tools based on a structured data model. An EMF model is named as Ecore and defines a meta-model language that can be implemented using EMFatic. EMFatic is a language used to represent EMF Ecore models in a textual form and has the *.emf extension. The EuGENia tool works as a front-end for GMF (Graphical Modeling Framework) and facilitates the Ecore models handling.

Thus, it was possible to generate the editors in Java class for the EMF meta-models that represents user story and i* model. Then the object-oriented structures were used based on the transformation between the models to perform mapping and implementation of the heuristics. The packages *.us2star.us and *.us2star.istar have interfaces and enumeration classes. The packages *.us2star.us.impl and *us2star.istar.impl have the implementation classes related to interfaces.

To implement the tool, we separate in a set of stages. In the first stage, we develop the user story metamodel using the Emfatic language. In addition, we used the meta-model of the i* model based on Paes [7]. From the .emf files were generated .ecore extension files. With the .ecore files and EuGENia tool a structure that represents the meta-models of both user story and i* are generated. The US2StarTool is an executable file based on java environment, named "US2StarTool.jar", and doesn't require any installation process.

The US2StarTool requires to the user an input file, in XLS format, that has been imported by an external tool responsible for managing user stories. If the user does not have this file, it will have to access this tool in order to get it and will upload it in the US2StarTool that will map to imported user stories. Then the US2StarTool generates an output file in XMI format capable to be imported by an i* model editor, so that it generates the graphical representation of this model.

The *.reader.xls package is responsible for obtaining the input file with *.xls* extension, and interpret it in order to read the user stories. Then, the package *mapping.us can communicate to obtain user stories and create objects from OO structure, that represents the user story model. This structure is composed by packages in *.us. After build the user stories objects, the mapping process is performed, using a set of heuristics. Thus, the package *.mapping.istar communicates with the packages *.mapping.istar.command, that contains the implemented heuristics, and *.mapping.us that contains the objects from user stories, and perform the mapping, creating OO objects structures that represents the i* model, composed by packages in *.istar. Finally, the package *.writer.xmi communicates with *.mapping.istar, which contains objects to create elements of output file in the *.xmi format.

The set of mapping heuristics were implemented based on the proposed heuristics from Jaqueira [5]. The user stories are imported into US2StarTool, resulting in the creation of objects represented by the elements in the model of user stories. These models are stored by UsData class, localized in the package *.mapping.us. Therefore, the heuristics are able to map elements that represents the i* model and keep in memory by the IstarData class, localized in the package *.mapping.istar. The mapping between models are performed through the UsData2IstarData class that reads the data in UsData and safe the objects in the IstarData. The heuristics are presented at Table 1 according to the sequence in which they are executed during mapping. Were used the concepts and notations according to i * Wiki [4] that is a simplified version of the technique.

Table 1. Proposed mapping heuristics from Jaqueira [5].

ID	**Description**
SD-H1	Create the System Actor;
SD-H2	Create an actor in i* model for each different role of user stories;
SD-H3	Create a goal in i* model for each goal of user stories. If there are repeated goals they will be defined only once in the model;
SD-H4	If there are repeated goals for different actors, create a generic actor;
SD-H4.1	Create a IS_A relationship of the generic actor for other specific actors who share the same goal;
SD-H5	Relate the dependencies of each actor with his goals;
SR-H1	Create a task in the Actor System for each action of user stories;
SR-H2	If there are different actions for the same goal, create a generic task;
SR-H2.1	Decompose the generic task into subtasks that represent the actions associated with the same goal;
SR-H3	Relate the dependencies of each goal with the corresponding tasks according to user stories;
SR-H4	If there are tasks that depend on itself actor to which they are related, generated a resource to the task name;
SR-H5	Relate the resource created, depending on the actor.

To use the US2StarTool, it’s needed that user has the user stories to upload in the application. We perform an integration of US2StarTool with a tool based in the management of backlogs, named Easybacklog (EB). The EB tool is useful for agile development and customer teams, to manage user stories. Using this tool it’s possible export an excel file (*.xls), that contains the data related to the backlog registered in the EB tool, including the user stories, that will be mapped by US2StarTool. Then, the *.xls file is imported by US2StarTool.

In this work, we use the IstarTool as graphical editor to generate and export the i* model [7]. This tool is able to create i* diagrams, validating the syntax rules based on the official rules defined by [4]. To integrate the US2StarTool with IstarTool, it was generated a XMI file with the *.istar extension. The figures 1 and 2 presents the activities involved in the process using the US2StarTool.

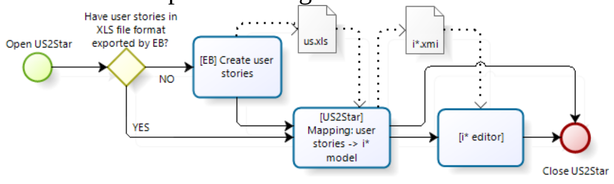

Open US2Star
Have user stories in XLS file format exported by EB?
NO
YES
[EB] Create user stories
us.xls
i*.xmi
[US2Star] Mapping: user stories -> i* model
[i* editor]
Close US2Star

Fig. 1. External process to upload user stories to be mapped by US2StarTool.

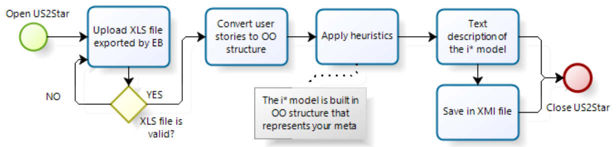
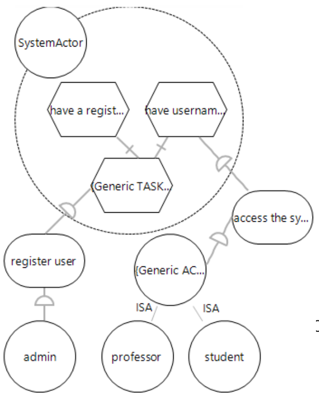

Open US2Star
Upload XLS file exported by EB
NO
YES
XLS file is valid?
Convert user stories to OO structure
Apply heuristics
The i* model is built in OO structure that represents your meta
Text description of the i* model
Save in XMI file
Close US2Star

Fig. 2. Showing the process of use of this Tool.

To demonstrate the operation of the US2StarTool, we used four user stories of a login system presented in Table 2 as an example. The i* model generated by US2StarTool is shown in Figure 3.

Table 2. Part of User Stories of a login system.

US01	**As** professor **I want** have username and password **To** access the system	**US03**	**As** admin **I want** have a registration request **To** register a user
US02	**As** student **I want** have username and password **To** access the system	**US04**	**As** admin **I want** have username and password **To** register a user



US2 ɔol.

4 Scientific contributions

Visual models assist in understanding of how users will need to use the system and are effective for the stakeholders to understand the proposed solution and also to keep them interested and involved [1]. Even in an agile environment it is necessary to develop some models before any implementation to ensure a shared understanding by the development team. Sharp et al. [8] concluded that the user stories are limited artifacts to provide understanding of the system as a whole and their dependency relationships are omitted.

The most important contribution of this work is the implementation of the tool named US2StarTool that maps a set of requirements described by user stories to a graphical model using the i* technique.

Therefore, the tool can assist the requirements engineering in the agile methodology context, contributing to a better view of actors and their dependencies in the system; better knowledge about actor roles inside the system-to-be; understand how actors can achieve their goals and actions; prioritize actors context and their dependencies in the system-to-be; define related functionalities and assign them to the same team.

5 Conclusions

The implementation of a tool to get i* models from user stories was successfully performed. The construction of US2StarTool enabled the study of requirements representation models in agile methods and i* technique. Were also studied the technology for creation of metamodels in Emfatic language and the use of EuGENia tool, both to generate the corresponding structure in Java to create metamodels, and for the creation of the graphic editor of the i* model. Heuristics mapping have been implemented and integrated with the Easybacklog tool to assist in creating the user stories, and finally, generate an input file for the graphics editor IstarTool. An example of application using real data (available at www.dimap.ufrn.br/~marciaj/US2StarTool) allows us to test the tool according to the analysis of the results suggesting its usefulness and contributions in agile development environment.

6 Ongoing and future work

A limitation of the tool is that is not possible to display the Resource element in the i* model generated, because as from a model of user stories is not possible to obtain this information to be mapped. One solution would be user interaction with US2StarTool to inform the dependencies between tasks and actors. The user can enter this information in the file .xmi exported by US2StarTool, or in the i* model editor, the IstarTool. Another limitation is the fact that it was not possible to generate a graphical representation of the i* model in itself US2StarTool. This occurred because the i* editor uses GMF technology, ie, is dependent on the Eclipse platform.

In order to continue the research for this work a few suggestions of further work can be cited. Improve the mapping process in order to display the resources in the i* model. User interaction with the US2StarTool after file import containing user stories is required. Thus, the user can tell whether action of the user stories depends on a specific feature of an actor. Integrate the tool with a graphical editor i* without the need to export a file and use another tool for this purpose. The i* model would be shown in the own US2StarTool. This is not possible because the graphics generators used are dependent on the Eclipse platform, since they are based on GMF. View the process of action of heuristics step-by-step at the time of mapping, in order to facilitate the user who wants to learn about the mapping process. Show the result of mapping for SD and SR models separately. The tool displays the SR model of the i*, which is an evolution of the SD model with the expansion of Actor System. Developing the reverse mapping, or mapping model for user i* stories. Write a manual for using the tool.

References

1. Beatty, J. e Chen, A. Visual Models for Software Requirements. Washington, Microsoft Press: 2012.
2. Cohn, M.: User Stories Applied: For Agile Software Development (The Addison-Wesley Signature Series), March. Addison-Wesley Professional, Reading (2004).
3. Gomez, A., Rueda, G., & Alarcón, P. P. (2010). A Systematic and Lightweight Method to Identify Dependencies between User Stories. In A. Sillitti, A. Martin, X. Wang, & E. Whitworth (Eds.), Agile Processes in Software Engineering and Extreme Programming (Vol. 48, pp. 190-195): Springer Berlin Heidelberg.
4. http://istar.rwth-aachen.de/tiki-index.php?page=i*+Wiki+Home. Acessado em: Maio de 2015.
5. Jaqueira, Aline; Lucena, Márcia; Aranha, Eduardo; Alencar, Fernanda and Castro, Jaelson. Using i * Models to Enrich User Stories. Proceedings of the 6th International i* Workshop (iStar 2013), CEUR Vol-978. 2013.
6. Kolp, Manuel, Paolo Giorgini, and John Mylopoulos. "A goal-based organizational perspective on multi-agent architectures." Intelligent Agents VIII. Springer Berlin Heidelberg, 2002. 128-140.
7. Paes, J. AGILE: Uma Abordagem para Geração Automática de Linguagens i*. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de Pernambuco, 2011.
8. Sharp, H., Robinson, H. Segal, J. and Furniss, D. (2006) 'The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective', Proceedings of Agile 2006, IEEE Computer Society Press, pp 65-75.
9. Yu, E. "Modelling Strategic Relationships for Process Reengineering". PhD thesis.University of Toronto, Department of Computer Science. 1995.