iStarTool: Modeling requirements using the i* framework

Átila Malta[1], Monique Soares[1], Emanuel Santos[1], Josias Paes[1], Fernanda Alencar[2], Jaelson Castro[1]

[1] Universidade Federal de Pernambuco – UFPE, Centro de Informática, Recife, Brazil
{avmm, mcs4, ebs , jpsj2, jbc}@cin.ufpe.br
[2] Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernandaalenc@gmail.com

Abstract. The iStarTool supports the graphical modeling of i* Framework. With a view to decrease the learning curve of i* models as well as to improve their quality, we provided the syntax checking feature. The tool allows the construction of valid models according to constraints and good practices guidelines. It is been developed using the open-source Eclipse platform and model-driven technologies, such as the Graphical Modeling Framework (GMF).

Keywords: iStarTool; Modeling tool, i* (iStar) Framework

1 Introduction

IStarTool is a graphical editor for the i* Framework models, built using the GMF framework. It supports the creation of both the SD and SR models, through an intuitive user interface built over the well-known Eclipse platform.

Download Information/availability. The iStarTool is free for download, and it is available at the project Web page: http://portal.cin.ufpe.br/ler/Projects/IStarTool.aspx.

Web page and documentation. The web page of the project is the same of the download page. The main documentation available in [1], is written in Portuguese but currently under translation to English.

Main purpose. The main purpose of the iStarTool is to facilitate the learning of i* language and improve the quality of i* models, being especially aimed at beginners. In several occasions during the teaching of i*, we identified the need to have a tool to help beginner users to reduce their number of mistakes well as to make it clear what kind of constructions were possible (or not). Several tools can handle some level of constraints over the i* models. In some cases also allowing, if necessary, the definition of extra modeling constraints, e.g., jUCMNav. However, we preferred to develop our own solution instead of customizing somebody else tool. The reasons are many fold. It served as an exercise to master the meta-modelling technologies and implementation environment. Furthermore, it allowed us to improve its usability, i.e., as we managed to provide more friendly warnings. In the future it will also incorporate model-driven features (see future works).

Status and Maturity. The current version of the iStarTool (Fig. 1) is 0.3. Some bugs still need to be fixed before the release of version 1.0. However, at this early stage, the tool presents a good degree of maturity and stability.

2 iStarTool

The iStarTool supports two different versions of i* Framework: the Yu's PhD Thesis and the iStarWiki [2]. Fig. 1 shows a screenshot of the tool interface. Panel 1 is where we can create and edit the diagrams. Panel 2 is the tool palette that offers the elements and links used when drawing diagrams. Panel 3 shows an overview of the diagram that allows the manipulation of large and complex diagrams. Panel 4 enabled the change of the properties of a selected element. Panel 5 highlights the button used to check (syntactically) the model.

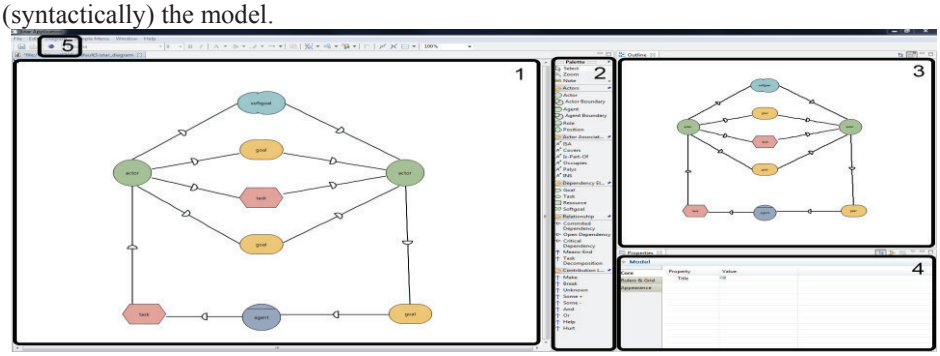

Fig.1. Screenshot of iStarTool

Primary Features. One of the main features of the iStarTool is the syntax checker. Due to the fact that we have used GMF, which supports the Object Constraint Language (OCL), it was possible to handle many common errors and constraints present in the i* models. This syntax checker feature consists of two independent systems integrated to the iStarTool, namely the the Syntax Warnings and the Syntax Checker System, described below.

The Syntax Warnings is an online system that enable pop-ups that fires warnings when the modeler makes a mistake during the modeling process. Instead of forbidding errors in the i* models, we prefer to alert users when some error is detected. For example, if a link is used in the wrong place, the Syntax Warning System shows a pop-up with an explanation, indicating to the user what is inappropriate. Then the user can correct it and learn in this process of trial and errors.

This system was deveopled because i* language allows some conditions that are difficult to be expressed and handled using OCL. Our systems can detect some of the most common problems and adequately treat them without upsetting the user. For example, if the user tries to link two actors without determining the intentional element an alert is fired. Many common mistakes are documented in the literature, see for example the i* good practices guidelines available at [2]. The errors detected include:

- Actors linked by dependency link without an intentional element (SD Model);

- Dependency links used inside an actor boundary (SR Model) linking internal elements;
- Contribution, Means-end or decomposition links used between actors in SD Models;

Unlike the Syntax Warning System, the Syntax Checker runs offline. The reason it that some i* steps could be considered wrong if analyzed in real time. In order to execute it, the user only needs to click a button (see Fig 1, panel 5). The Syntax Checker detects the same problems of the Syntax Warning and many others such as:

- The same intentional element being targeted more than once;
- The same intentional element being the source more than once;
- Dependency links without an intentional element in a SR model.

The iStarTool also allows users to work on multiple models at the same time. It saves all models in XML format, by default, and if necessary, they can be exported as an image. In a previous work we have identified the common and variable modeling elements of several i* based languages [4]. Based on SPL principles, we produced a core metamodel, which enabled the definition of specific i* dialects. Moreover, the current features of the iStarTool can be reused to support families of graphical editors for i* based languages. Since the AGILE approach was based on the iStarTool, similar families of tools could be generated in minutes.

3 Limitations and Future plans

We are planning to make several extensions. For example we want to support the measurement of i* models (based on some well pre-defined metrics) as well as to provide some semantic checkers present. We plan to integrate model transformations to produce other artifacts such as architectural models and scenario descriptions [5].

We intend to improve the interface of the iStarTool as well as the shape of the i* elements. Moreover, we wish to provide the interoperability with other i* modeling tools. Currently, the iStarTool just runs on Windows. However we are already working to support other operational systems.

References

1. Santos, B. S. IStarTool – A proposal of tool for modeling i*. (in portuguese, Uma proposta de ferramenta para modelagem de i*). Master's thesis - Universidade Federal de Pernambuco, Centro de Informática, Brasil, 2008.
2. i* Wiki: i* Guides < http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=200 >. Last Access in May of 2011.
3. Graphical Modelling Framework < http://www.eclipse.org/modeling/gmp/ >. Last Access in May of 2011.
4. Paes, J., Lima, , Santos, E., Silva, C., and Castro, J.: AGILE : Automatic Generation of i * Languages. In: Proceedings of the 24th Ibero-American Conference on Software Engineering - CIbSE 2011, Rio de Janeiro, Brasil: 2011.
5. Lucena, M., Silva, C., Santos, E., Alencar, F., and Castro, J.: Applying Transformation Rules to Improve i* Models. In: Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009), Boston, USA: 2009, pp. 43-48.