H*i*ME: Hierarchical *i** Modeling Editor

Lidia López [1]
Xavier Franch [1]
Jordi Marco [1]

Abstract: In this paper, we present H*i*ME, a tool for editing *i** models. The distinguishing characteristic of H*i*ME is its ability to deal with inheritance. It includes specific operations for declaring an actor as heir of another and then stating the relationships between the intentional elements of both actors.

1 Introduction

Inheritance is defined in the *i** framework using an "is-a" link between actors. Although this construct already appeared in the seminal version of the framework (1995) [1], its semantics has not been completely defined neither in that original version nor in other existing *i** dialects. Since we had the need of defining completely the meaning of inheritance, in earlier works [2, 3] we explored its definition, emphasizing on how the is-a relationship between actors affects to their intentional elements (IEs).

A natural consequence is the need of dealing with inheritance in *i** related tools. There are several tools which support the seminal Yu's proposal and *i** dialects [4]. As a natural consequence of the lack of precise definition of the notion of inheritance, tools like OME, REDEPEND and TAOM4E do not include specific functionality for inheritance, at most they support the establishment of is-a links. Our aim thus is to provide tool support for helping modelers to use inheritance according to [2, 3].

2 Inheritance in *i** Models

We have defined inheritance for actors and how the inherited IEs, in the heir, can be modified. When an actor (heir) is linked by an is-a relation with another actor (parent), the heir inherits all the parent's IEs. Upon this basic behavior, we have defined three different operations over IEs that, under certain conditions, lead to correct heirs: refinement (specializes an IE), extension (adds some information about how to obtain the IE) and redefinition (defines a different way from the parent to obtain the IE). In [2] the reader may find the complete definition of these three operations: their context, their correctness conditions and the effects on the heir.

[1] Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3 (Campus Nord, Omega building), 08034 Barcelona, Catalunya, Spain, {llopez,franch,jmarco}@lsi.upc.edu

3 Hierarchical *i** Modeling Editor

We aim to provide a tool that includes the inheritance operations introduced in the previous section. For this purpose we had two options: creating a new modeling tool from scratch or including new operations in an existing one. We have opted for the second alternative, and have chosen the J-PR*i*M tool [5] as starting point, not just because it has been developed by our research group, but also because the modeling facilities may be decoupled very easily from the core functionality of the tool, namely the support to the PR*i*M methodology for reengineering business processes. Therefore, we have split the edition functionality in a separate tool and then included the inheritance operations only for edition purposes. We call H*i*ME (for Hierarchical *i** Modeling Editor) the resulting tool.

The current version of H*i*ME (1.0) allows creating and managing models using the general functionality that can be found in most *i** existing tools (creating actors, new IEs for an actor, new dependencies, etc). H*i*ME also imports and exports *i** models using iStarML [6]. One feature that distinguishes H*i*ME from other similar tools is that the model is not represented graphically following the symbols of the *i** framework; it is represented like a folder tree directory in a file system. **Figure 1** shows a piece of the Meeting Scheduler example found at [1].

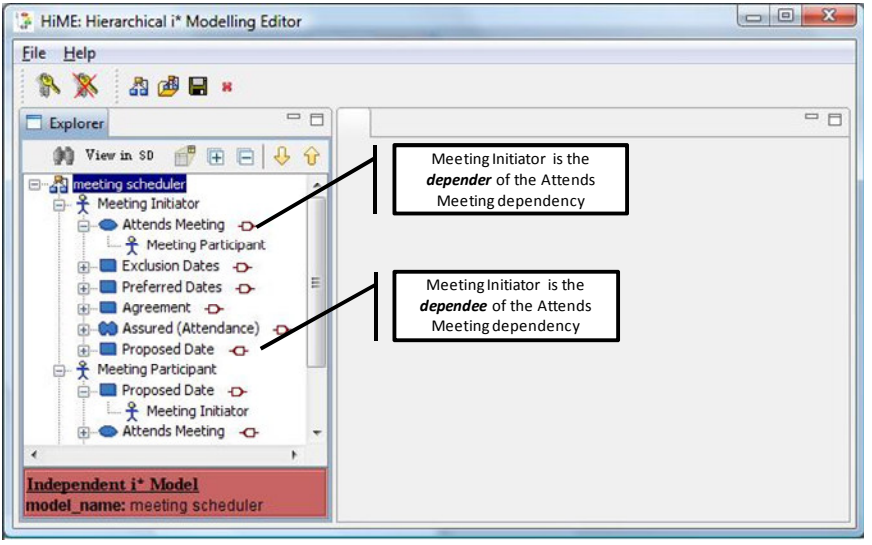

Figure 1. Meeting Scheduler model (SD) as represented in H*i*ME

The functionality added to support inheritance is that an actor can be created as an heir of another, which implies the inclusion of an is-a link between them. Heirs inherit all intentional elements from their parents, and the inherited elements can be modified in the heir using the operations described in the previous section. All parent's IEs are inherited, but

only the modified ones (using the inheritance operations) and the new ones appear inside the boundary of heir actors. This characteristic helps in limiting the complexity of the obtained models and allows focusing in incremental changes. We are currently implementing a functionality that will allow the user viewing all heir's IEs (making visible all inherited IEs). **Figure 2** shows the valid operations: adding a subactor (left); and refinement, extension and redefinition of intentional elements (right).

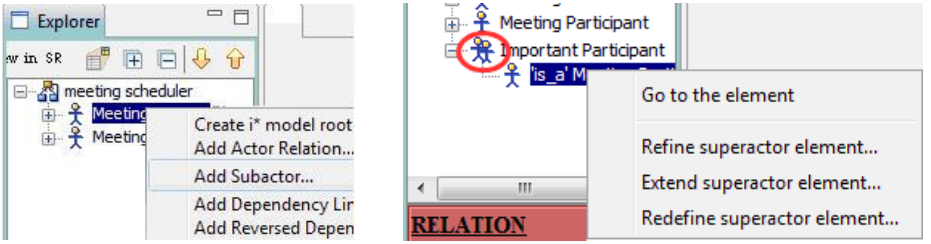

Figure 2. Inheritance operations and the effect of adding an heir

Therefore, the tool shows 2 kinds of IEs in heirs, the inherited IEs from the parent and the new IEs from the heir. To distinguish among them, H*i*ME uses regular colored icons next to the IE name to identify the type of the IEs that are defined as new in an actor, whilst these icons are shown in black and white for inherited IEs. **Figure 3** shows an example, emphasizing these two different types of IEs for the Important Participant heir.

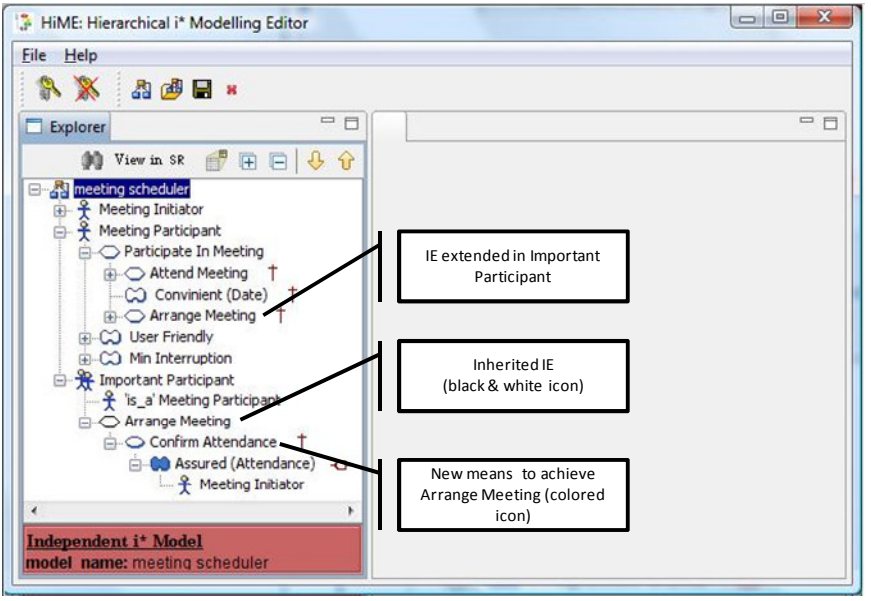

Figure 3. Inherited IEs for the Important Participant actor in H*i*ME

The Hierarchical *i** Modeling Editor has been developed using Java and the Rich Client Platform (RCP) for Eclipse. Models are stored in a MySQL database.

The tool is available in http://www.lsi.upc.edu/~llopez/hime/. Besides the necessary files for the tool installation (RCP files and the script to create the database), a user manual is also available.

4 Future Work

Future work moves along two directions. On the one hand, we are working to improve functionality. In the next versions, we will address two current limitations of the tool, transitivity and multiple inheritance. Also, we plan to include an inheritance operation for dependencies between actors and actor types (agents, positions and roles). For dependencies, inheritance operations will be defined to refine the dependum and the dependency strengths. Also, correctness conditions checking (i.e., operations are applied under appropriate circumstances) are planned for the future releases.

On the other hand, concerning tool architecture, we plan to eliminate the need of storing models in a data base. Next versions will store them in a file, making models portability possible and facilitating the installation process (no DBMS required).

5 References

[1] Yu, E. *Modeling Strategic Relationships for Process Reengineering*. PhD Thesis, Univ. Toronto, 1995.

[2] Clotet R., Franch X., López L., Marco J., Seyff N. and Grünbacher P.: The Meaning of Inheritance in *i**. In *Proceedings of the 17th International Workshop on Agent-Oriented Information Systems* (AOIS'07), Tapir Academic Press, Trondheim, Norway, June 2007.

[3] López L., Franch X., Marco J.: Defining Inheritance in *i** at the Level of SR Intentional Elements. In *Proceedings of the 3rd International i* Workshop*, CEUR Workshop Series 322, Recife, Brazil, February 2008.

[4] *i** Wiki site: http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=21.

[5] Grau G., Franch X., Ávila S. J-PRiM: A Java Tool for a Process Reengineering *i** Methodology. In *Proceedings of the 14th IEEE International Requirements Engineering Conference* (RE'06), IEEE Computer Society, Minneapolis, USA, Sept. 2006.

[6] Cares C., Franch X., Perini A. and Susi A.: iStarML Reference's Guide. *Technical Report LSI-07-46-R*, UPC, Barcelona (2007)