# A Complete Definition of the Inheritance Construct in *i**

Lidia López

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

**Abstract.** The is-a relationship among actors has been introduced in the *i** framework since its definition. However, its effect at the level of intentional elements and dependencies is not always clear. The main goal of this thesis is presenting a complete and non-ambiguous definition of inheritance for the i* framework. With this aim, it is necessary to define the modelling operations that make use of inheritance, explain how inheritance affects *i** treatments and properties, and how the inherited elements can be represented graphically.

**Keywords: *i** framework, inheritance, agent-oriented modeling**.

## 1 Introduction

Goal- and agent-oriented models are widely used in many fields like business process modeling, requirements engineering and others. The *i** framework is one of the most widespread approaches supporting these paradigms. It focuses mainly on representing strategic concerns by means of intentional elements and their relationships.

Inheritance is defined in *i** 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.

## 2 Proposal and Research Method

Due to the lack of formalisation about the use of inheritance in the *i** approach, the purpose of my thesis is "*Presenting a complete and non-ambiguous definition of inheritance for the i* framework*". This raises the following research questions:

- Which are the modelling operations that make use of inheritance?
- How does inheritance affect *i** treatments and properties?
- How can we represent graphically the inherited elements result of inheritance operations?

To provide a comprehensive approach to inheritance in *i**, I need to provide the semantics and syntax of inheritance.

For accomplishing the thesis objective, I have identified some milestones.

Regarding the semantics definition I need to:

(1) define i* models in a formal way (algebraic-based);

(2) define the rules which provide the inheritance definition (a list of rules and forbidden "situations");

(3) include inheritance in the i* metamodel; and

(4) explore how inheritance affects typical i* properties and treatments (used to analyse models).

For the syntax definition, I need to:

(5) define the rules for representing inheritance graphically;

(6) provide a tool for supporting inheritance modelling; and

(7) define the needed tags for iStarML (a Mark-up Language for *i** recently defined to share models between *i** tools [2]).

I have grouped all these tasks in a 3-phase planning. These phases were preceded by the problem identification and the study of the related work. In Table 1, the tasks mentioned above are distributed into the different phases of my research. The last phase consists on consolidating my research work by means of a complete validation of the proposal, as well as the thesis writing and final documentation.

**Table 1. Research Planning**

| | ***Phase 1*** | ***Phase 2*** | ***Phase 3*** |
|---|---|---|---|
| *Semantics* | Rules definition | Rules formalisation<br>Metamodel extension<br>Treatments and rules analysis | Packaging |
| *Syntax* | Rules definition | | Packaging |
| *Tool* | Existing tools analysis<br>Tool specification | Tool design<br>iStarML extension development | Packaging |
| *Validation* | Case studies definition | Case studies execution | Packaging |

## 3 *i** Inheritance Basis

The is-a relationship has been used by several teams in several contexts, in the same way as Yu did in his PhD thesis, see [3] and [4] for some illustrative examples. The reference model presented in [5] does not include inheritance. The main *i** dialects GRL and Tropos do not include inheritance as part of their approach neither. A comparative analysis of *i** dialects can be found in [6].

To avoid defining inheritance from scratch, I have taken the object-oriented (OO) paradigm as the basis of my proposal. OO inheritance consists on grouping all common functionality in a generic class. Classes that have this functionality (descendants) inherit from the general one (ancestor). The common functionality included in the ancestor class is not present on the descendants, but the descendants inherit it from the ancestor.

## 4 Research Performed

At this point, I am developing the second phase of my research. All information about the state of the art (corresponding to the previous research), the tool existing analysis and the validation cases definition were presented in my thesis project on June 2007.

An initial work about the rules definition (semantic and syntax) was presented in [7] and [8]. These papers presented the semantic and syntax rules for the modifications allowed on the inherited intentional element for heirs. When one actor (subactor[1]) is linked to another (superactor) using the is-a link, I have identified 3 operations that can be performed on superactor's IE for having the subactor's IE. For this operation identification, I have used the "*Taxomania rule*" from the OO paradigm, where "taxomania" is the contraction of "taxonomy" and "mania". It states that: "*Every heir must introduce a feature, redeclare an inherited feature, or add an invariant clause.*" [9] (pg. 820). These operations are:

- *Extension*: a subactor can add its own intentional elements and link them to the existing or new ones.
- *Redefinition*: a subactor can redefine an inherited IE. Inheritance must not change the semantics, so it is only in the way to achieve this IE.
- *Refinement*: a subactor can change the IE's meaning. It is only allowed when the IE meaning on subactor is a specialisation of the IE superactor.

This is an initial set of operations, a deeper revision of bibliography (e.g., [10]) is still ongoing to get the definitive set of operations. These operations were presented along with their syntax. Syntax is an important issue for *i** because of its highly graphical nature. Due to the usual complexity of *i** models, the inheritance syntax will be defined thinking about information economy, trying to add as few shapes and lines as possible to models. The general idea is including only the new IEs in subactors, the new elements will be used and shown as normal IEs. But, when the new IEs are linked to superactors' IEs, these ones appear inside the subactors' boundary in a different style (we have chosen to represent those using doted lines).

I already have some work done in phase 2, mainly the model and operations formalisation. This formalisation corresponds to a model definition in an algebraic style and the definition of rules to ensure model correctness. Now, I am formally proving that the operations definition, with the proper restrictions, maintain the model correctness. This verification consists on proving that using the defined operations, subactor's satisfaction implies superactor's satisfaction. For proving that, I defined the satisfaction of an actor as the satisfaction of all its objectives. I also have an initial proposal for including the inheritance in the metamodel presented in [5]. A new class to represent instances is needed in order to avoid inheritance between them.

Regarding the tool, there was not a complete specification at the end of the first phase. The reason is that I am not developing a tool from scratch; I am adding inheritance functionality to an existing one. After the study of the existing tools, I decided to add inheritance functionality in the editor component of the J-PR*i*M tool [11]. The result is a new editor that we call H*i*ME (Hierarchical *i** Modeling Editor) [12]. My specification-design-development process is iterative, for each new

[1] I have adopted the "subclass" and "superclass" terms used in the OO paradigm.

operation that I am including on the tool. At this point there are the specification, design and development for the 3 operations previously defined. News tags on iStarML are also included incrementally at the same time the tool grows up.

The remaining research for second phase is: completing the proof of correctness of model definition, analysing the inheritance behaviour for properties and treatments, and finishing the tool development.

## 5 Consolidation and Validation

Besides the formal definition verification, I also need to validate the viability of my proposal. For the validation, I am going to execute the case studies defined in the first phase. These cases are oriented to academic research. I have chosen two of the most popular examples used in *i** community: the meeting scheduler, presented in Yu's thesis, and a conference management system, widely known by the multi-agent systems community. Whilst running these examples using inheritance by myself, I would like to collect information about my proposal from two separate groups of users:

- Non-expert users. I will supervise some PhD/Master students that will develop some examples that need inheritance. Working with this kind of users, I will be able to know if my proposal is easy to learn.
- Expert users. The *i** community created the *i** wiki [13], where there is an *i** quick guide, information about events, publications, tools, etc. I will use this wiki to publish my proposal and invite *i** experts to give me some feedback.

For both user groups I will also ask them to answer a survey previously designed to gather their impressions. At this stage, I do not want to get opinions only for the proposal, but also about the tool. For this reason, the survey will contain questions related with the proposal and the tool features.

Even succeeding in collecting this information, for having a complete validation I need to test this proposal in a real case. I can take advantage of the fact that the research group I belong to, is involved in agreements with companies for collaborating in their projects (the group has followed this schema in previous PhD thesis) (see [14] for a recent case study). On the other hand, we are going to contact with a significant part of the *i** community that is using *i** in industrial experiences.

## 6 Contributions

The expected results of my thesis are the formal definition for inheritance (verified and validated) and a tool supporting all inheritance functionality defined.

Apart from testing the inheritance on academic and industrial case, I want to identify some areas where inheritance can be useful. I missed the inheritance definition when I was involved in a research project on multi-stakeholders distributed systems modeling: we found that inheritance helped us to model variability [15].

## References


1. Yu., E.: Modelling strategic relationships for process reengineering. Ph.D. dissertation, Univ. Toronto, 1995.
2. Cares C., Franch X., Perini A., Susi A.: "iStarML Reference's Guide". *Technical Report LSI-07-46-R*, UPC, Barcelona (2007).
3. Mouratidis H., Jürjens J., Fox J.: "Towards a Comprehensive Framework for Secure Systems Development". In *Proceedings of the 18th Conference on Advanced Information Systems Engineering* (CAiSE). Luxembourg June 2006.
4. OME tool website: http://www.cs.toronto.edu/km/ome/. Last accessed June 2009.
5. Cares, C., Franch, X., Mayol, E., Quer, C.: "A Reference Model for *i*"*. Book chapter in *Socio-technical systems specification in i**, E. Yu (ed.), The MIT Press, in press.
6. Ayala, C., Cares, C., Carvallo, J.P, Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., Quer, C.: "A Comparative Analysis of *i**-Based Agent-Ariented Modeling Languages". In *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering* (SEKE). Taipei, Taiwan, July 2005.
7. Clotet, R., Franch, X., López, L., Marco, J., Seyff, N., Grünbacher, P.: "The Meaning of Inheritance in *i**". In *Proceedings of the 17th International Workshop on Agent-Oriented Information Systems* (AOIS) *at* CAiSE. Trondheim, Norway, June 2007.
8. 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.
9. Meyer B.: *Object-Oriented Software Construction*. Prentice Hall, 1997.
10. Guizzardi, G., Wagner, G., Guarino, N. van Sinderen, M.: "Ontological Well-Founded Profile for UML Conceptual Models". In *Proceedings of the 16th International Conference Advanced Information Systems Engineering* (CAiSE'04), LNCS 3084, Springer-Verlag, Riga (Latvia), 2004.
11. **G**rau, 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), IEEE Computer Society, Minneapolis, USA, Sept. 2006.
12. López, L., Franch, X., Marco, J.: "Hierarchical i* Modeling Editor". To be published in *Revista de Informática Teórica e Aplicada* (RITA), 2010.
13. *i** Wiki website: http://istar.rwth-aachen.de/tiki-index.php.
14. Carvallo, J.P., Franch, X.: "On the use of *i** for Architecting Hybrid Systems: A Method and an Evaluation Report". To be published in *Proceedings of the 2nd Working Conference on the Practice of Enterprise Modeling* (POEM'09), LNBIP, Springer-Verlag, Stockholm (Sweden), 2009.
15. Grünbacher, P., Dhungana, D., Seyff, N., Quintus, M., Clotet, R., Franch, X., López, L., Marco, J.: "Goal and Variability Modelling for Service-oriented Systems: Integrating *i** with Decision Models". In *Proceedings of the Software and Services Variability Management Workshop*, Helsinki, April 2007.