Definition and Uses of the *i** Metamodel[1]

Carlos Cares[1,2], Xavier Franch[1], Lidia López[1], Jordi Marco[1]

[1]Universitat Politècnica de Catalunya, Omega-122, 08034 Barcelona, Spain
{ccares, franch}@essi.upc.edu, {llopez, jmarco}@lsi.upc.edu
[2]Universidad de La Frontera, Av. Francisco Salazar, 01145 Temuco, Chile

Abstract. The clear definition of a metamodel can be considered helpful for any conceptual modeling approach, and the *i** framework is not an exception. Agreeing on a metamodel for *i** can be considered even more convenient than ever when we are aware of the different dialects and variations that the community proposed, and keep proposing, over the seminal *i** definition. In this paper we present the revised version of the *i** metamodel proposed by the GESSI research group at 2005 and we report some current contexts of use: 1) definition of a data interchange format; 2) definition of the inheritance construct; 3) definition of a modularity construct; and 4) definition of a metrics framework.

Keywords: Goal-Oriented Requirements Engineering, *i**, Metamodel, iStarML.

1 Introduction

Since it was first released, the *i** framework has been adapted to the needs of specific research groups that wanted to represent concepts specific of their software engineering problem, like security, law compliance, trust modelling, architectural design, model-driven development and agent-orientation, among others. Even, the *i** framework itself has experienced a natural evolution that has led to a slightly modified version available in the *i** wiki. This set of main *i** variations have been object of our study which we have described and genealogically analysed in [1].

This diversity, although not necessarily pernicious, has some consequences. When reading a work around the *i** framework, it is necessary first to understand what concrete version of *i** is being used. If the contribution is based on the original framework, sometimes the authors declare which version are using (lately, it is happening to be the wiki version), but sometimes there is no explicit mention, which often makes the reader a bit hesitant about details of the proposal being presented. On the other hand, if the work is proposing some new variation, the semantics is sometimes given informally or by using a formalism which is not easy to align with the available descriptions of *i**. In order to deal with this problem we proposed at 2005 a reference metamodel for *i** [2] where particular metamodels of *i** variations can be obtained by applying UML refactoring operations. Since then, we have revised this metamodel upon which we have based our research work on: *i** inheritance, requirements interoperability, and metrics frameworks among others.

[1] This work has been partially supported by the Spanish project TIN2007-64753.

2 Objectives of the Research

In this report-of-progress paper, we review our proposal of metamodel for the *i** framework, we show our updated version and outline several contexts in which it has been used to formalize our *i**-related proposals. In particular, our main objective has been to define a metamodel able to express most of the current variations and extensions of the *i** framework and to use this metamodel as a reference model for our lines of research. More precisely, we have worked with four specific objectives in mind: (i) To define a metamodel for the *i** framework not bound to any particular technology, (ii) To use this metamodel as the underlying baseline for defining a *i** diagram interchange format, iStarML, (iii) To use this metamodel as a reference framework over which formulating our own extensions and variations of *i**, namely a full definition of inheritance and the concept of module, and (iv) To use this metamodel as the syntactic baseline over which formulating a framework for the definition of metrics on *i**.

3 Scientific Contributions

As a first tangible contribution, we are proposing an *i** metamodel compliant with the objectives stated in Section 2 (generality, flexibility, technological independence; see Figure 1). It has been built by consolidating several main versions of the framework, as thoroughly described in [1]. It presents some superclasses of interest (*Node* as the most general one, and also *DependableNode* and *IntentionalElement*) and then the most relevant *i** concepts: *Actor*, *SR-Element* and *SD-Dependum* as classes, and *Relationship* and *Link* as association classes, all of them with the appropriate subclasses.

The iStarML interchange format [3, 4] has being designed starting from the metamodel. The format implements the metamodel as an XML grammar and also supports the possibility of extension with new constructs. The ccistarml v0.6 Java package (http://www.essi.upc.edu/~ccares/ccsoftware/ccistarml_v0.6.1.zip) allows creating, importing and handling iStarML-compliant files. Several tools have been and are being customized to support importing and exporting iStarML. Remarkably, the H*i*ME tool (http://www.lsi.upc.edu/~llopez/hime/) supports both export and import, whilst we have developed an online translator from OME .tel files into iStarML (http://www.essi.upc.edu/~ccares/index.php?section=ometranslator). It is planned for adoption in a next release of TAOM4E (http://sra.itc.it/tools/taom4e/).

The inheritance proposal presented in [5, 6] and the module construct as defined in [7] have been related to the metamodel. This makes the definition of both concepts easy to integrate into the *i** framework. In the case of inheritance, the integration is very tight since inheritance appears in the metamodel itself, therefore we are just providing a more detailed definition of a core concept. The concept of inheritance has been implemented in the H*i*ME tool presented above, supporting the three operations identified in [5]: extension, refinement and redefinition. As for modules, the option has been to integrate in a loosely coupled way, meaning that modules are linked to the metamodel but the metamodel is not modified.

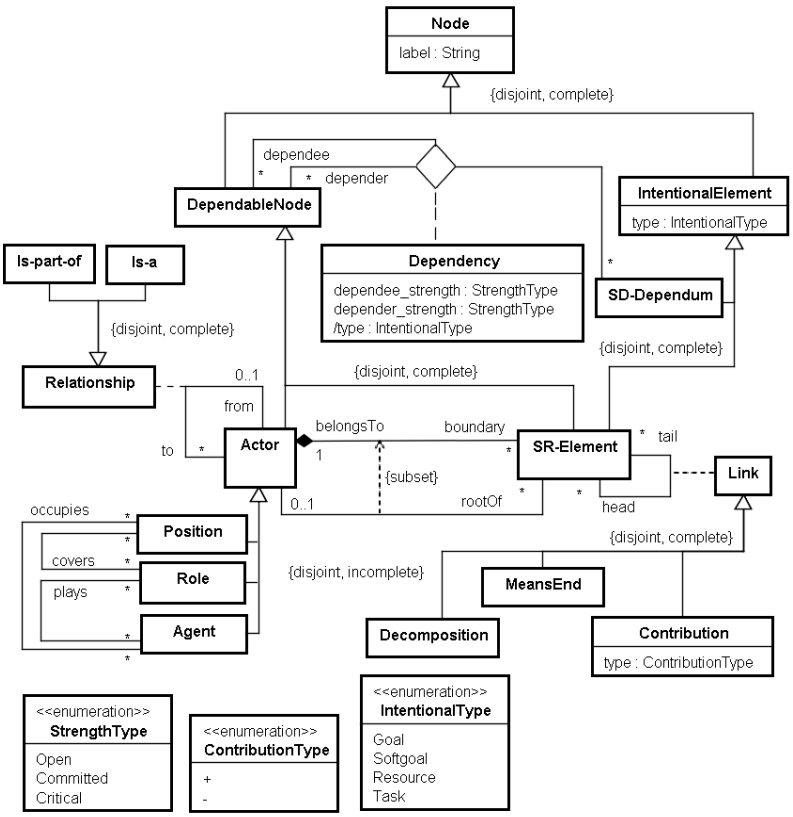

Fig. 1. The *i** metamodel.

Figure 2 provides an excerpt of both modifications. We may observe that in addition to actor inheritance, we are allowing the refinement of dependencies and intentional elements in general, and extension and redefinition only for SR-elements. Concerning modules, in addition to the general concept of module, we identify SD-modules and SR-modules. SR-modules may be of different types too, e.g. for storing means-end decompositions. New classes are introduced for these concepts, coupled to their counterpart elements in the metamodel.

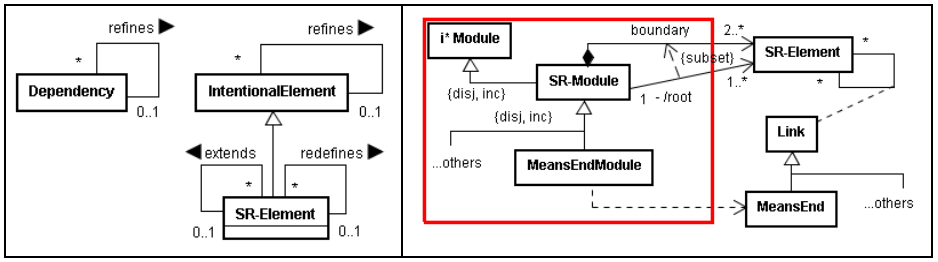

Fig. 2. Extending the *i** metamodel, excerpts: inheritance (left) and modules (right).

As a last remarkable use of the *i** metamodel, we have used it as the baseline for formulating an *i** metrics definition framework. In [8], we have provided a catalogue of metric definition patterns in which their form is expressed as an OCL template involving metamodel elements (see Figure 3 for an example). In [9], we have illustrated a particular case of application, the definition of a metric suite for business process modeling. In this exemplar we may observe the general procedure in which an extension of the *i** metamodel for capturing the essential concepts of the domain of interest (business processes in this case) is needed. Then, the metric suites that exist in this domain are mapped into their counterparts using a metamodel mapping.

Name	**Dependency-Based (Metrics Definition -> Quantitative -> Structural)**
Context	Some metrics have sense when applied to dependency links
Problem	The metrics will depend not just on the characteristics of the dependency link itself, but also on the two actors that act as depender and dependee
Solution	Identify three different factors that influence the metrics: one bound to the dependency link itself (probably related with the type of its dependum), and the others to the two actors, depender and dependee
Required knowledge	The effect of the depender, the dependee and the dependum in the metric, represented by three functions: filter: Dependum → Float correctionFactorDepender: Actor → Float, correctionFactorDependee: Actor → Float
Form	**context** Dependency::metric(): Type **let** ownerActor(x: DependableNode): Actor = **if** x.oclIsTypeOf(Actor) **then** x **else** x.owner **in**: **post**: result = self. dependum.filter() * ownerActor(self. depender).correctionFactorDepender() * ownerActor(self.dependee).correctionFactorDependee()

Fig. 3. Defining metric patterns by means of OCL templates over the *i** metamodel.

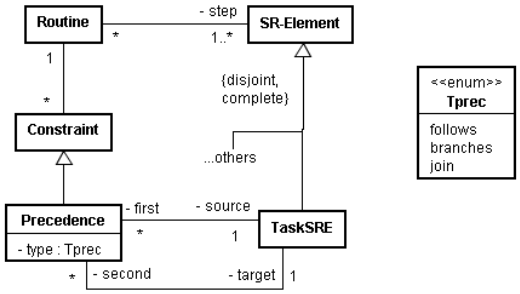

Fig. 4. Extending the *i** metamodel for defining metrics over business process models.

4 Conclusions

This main purpose of this paper has been twofold. On the one hand, illustrating the form that the *i** reference metamodel takes by including a particular proposal. On the second hand, providing an overview of the different uses of such a metamodel in different contexts that may be of general interest for the *i** community: for model interchange, for definition of new concepts and for definition of metrics.

Several authors agree on our belief that the existence of an *i** metamodel could bring some benefits (shared understanding, tool interoperability, etc.) to the *i** community [10, 11], although the statement could in fact be a matter of discussion, and in fact some other researchers advocate for more focused metamodels like the Tropos metamodel [12] and the GRL metamodel [13]. Our view is that the differences among the several existing approaches concerning the core concepts of *i** are not so severe as to prevent the proposed agreement, whilst the potential benefits seem attractive enough.

5 Ongoing and Future Work

We think that the most important future work is a community work: agreeing on a metamodel as the "official" *i** framework metamodel (being the one presented here or other), making it available in the *i** wiki for reference. Its existence shall provide a shared context to *i** researchers and practitioners, and shall serve as reference for: new extensions and variations, semantic and pragmatic agreements, tool support, etc.

Concerning our particular future work, we plan to advance in the following research lines: (1) using iStarML as the technological infrastructure to connect as many available tools as possible. This also means coping with the mapping problem where a construct that is used in some source tool is not supported in some destination tool. An example of how to deal with this case has been presented in [4]; (2) completing the definition of inheritance, providing the necessary restrictions on the use of the identified operations (extension, refinement and redefinition [5]) in the form of OCL constraints over the metamodel elements; (3) creating a comprehensive catalogue of metrics suite based on the use of the patterns identified in our previous work; (4) implementing the concepts presented here (inheritance, modularity and metrics) using our H*i*ME tool, and (5) providing semantics to the metamodel (i.e., how the different concepts proposed in the metamodel should be interpreted).

References

1. Cares, C., Franch, X., Mayol, E., Quer, C. "A Reference Model for *i**". In: *Social Modelling for Requirements Engineering*, The MIT Press, 2010 (in press).
2. Ayala, C.P., Cares, C., Carvallo, J.P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., and Quer, C. "A Comparative Analisys of *i**-Based Goal-Oriented Modelling Languages". AOSDM 2005.
3. Cares, C., Franch, X., Perini, A., Susi, A. "iStarML: An XML-based Model Interchange Format for *i**". In: *Procs. of the 3rd International i* Workshop*, CEUR-WS 322, 2008.
4. Cares, C., Franch, X., Perini, A., Susi, A. "Towards Interoperability of *i** Models using iStarML". *Computer Standards & Interfaces*, Elsevier (in press).
5. Clotet, R., Franch, X., López, L., Marco, J., Seyff, N., Grünbacher, P. "The Meaning of Inheritance in *i**". AOIS 2007.
6. López, L., Franch, X., Marco, J. "Defining Inheritance in *i** at the Level of SR Intentional Elements". In: *Procs. of the 3rd International i* Workshop*, CEUR-WS 322, 2008.
7. Franch, X. "Incorporating Modules into the *i** Framework". CAiSE 2010.

8. Franch, X., Grau, G. "Towards a Catalogue of Patterns for Defining Metrics over *i** Models". CAiSE 2008.
9. Franch, X. "A Method for the Definition of Metrics over *i** Models". CAiSE 2009.
10. Cabot, J., Yu, E. "Improving Requirements Specifications in Model-Driven Development Process". ChaMDE 2008.
11. Moody, D.L., Heymans, P., Matulevicius, R. "Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of *i** Visual Syntax". RE 2009.
12. Susi, A., Perini, A., Mylopoulos, J., Giorgini, P. "The Tropos Metamodel and its Use". Informatica, 2005.
13. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G. "A Lightweight GRL Profile for *i** Modeling". RIGiM 2009.