From *i** to OO-Method: Problems and Solutions

Fernanda Alencar[1], Beatriz Marín[2], Giovanni Giachetti[2], Emanuel Santos[1], Oscar Pastor[2], Jaelson Castro[1], Xavier Franch[3]

[1]Universidade Federal de Pernambuco, Av. Prof. Luiz Freire s/n, 50740-540, Recife, Brazil
fernanda.ralencar@ufpe.br, {ebs, jbc}@cin.ufpe.br
[2]Universidad Politécnica de Valencia, Camino de Vera s/n, CP:46022, Valencia, Spain
{bmarin, ggiachetti, opastor}@dsic.upv.es,
[3]Universitat Politècnica de Catalunya, Omega-122, CP: 08034, Barcelona, Spain
franch@lsi.upc.edu

Abstract. Nowadays, the successful development of software products depends on a good understanding of the system requirements. The *i** framework offers expressive models to capture social and intentional characteristics in an organizational context. However, there is a well-known gap between intentional *i** models and other conceptual models used for software development. In order to reduce this gap, we have developed a transformation process to obtain from *i** models an appropriate input for the OO-Method Model Driven approach. In this paper, we present the problems detected from the application of this transformation process and the possible solutions, which are oriented to improve the alignment of *i** and OO-Method conceptual models.

Keywords: Goal-Oriented Requirement Engineering, i*, Requirement transformations, OO-Method, Model-Driven Development.

1 Introduction

Currently, an appropriate requirement specification is a key aspect for the correct development of software systems [9]. Requirements specification should include not only software specifications, but also multiple complementary views: intentional, structural, behavioral, functional, presentational, etc.

Goal-Oriented Requirements Engineering (GORE) stood out because it is mainly concerned with the stakeholder intentions and their rationales. Among the several GORE works, we have chosen the *i** framework [17] because it is a consolidated modeling technique with good tool support [7], and an abstract syntax formalized by a metamodel specification [10].

Nonetheless, it is still an open question the relationship between the intentional models described in terms of i* and the remaining conceptual models (e.g. structural, behavioral, functional, presentational views) used in other well-known model driven approaches.

In this paper we report on lessons learnt with a collaborative project[1], which aims at relating i* and the OO-Method approaches. The *OO-Method* is used as a reference MDD technology because it has been successfully applied to industrial software development [14] by means of the *OlivaNova* suite [3].

This rest of this paper is organized as follows: Section 2 presents our approach. Section 3 presents some problems that have arisen in the application of this approach and the solutions proposed for these issues. Finally, section 4 presents our conclusions and further work.

2 Relating i* and OO-Method Approaches

We propose a transformation process presented with the Business Process Modeling Notation (BPMN [13]) and composed by two sub-process, i* Models Analysis and Transformation Guidelines (further details in [1] and [2]), to obtain an OO-Method class model from an *i** model (see Fig. 1).

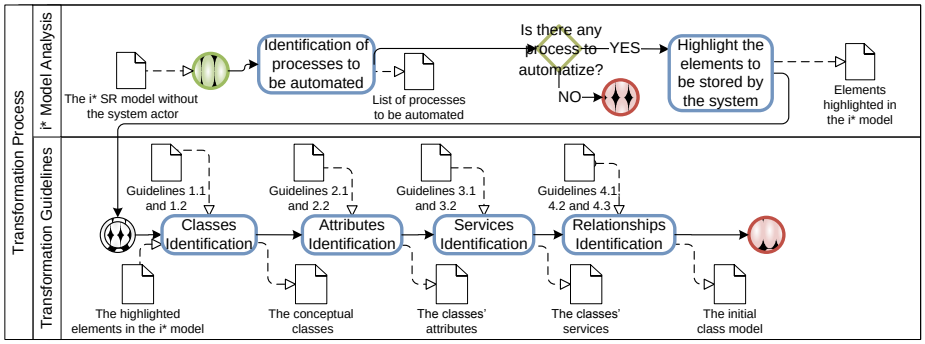

Fig. 1. The transformation process modeled with BPMN [13]

Initially, we analyze the goals defined in the Early SR model (see Fig.1, first activity: Identification of processes to be automated) to capture the organizational processes that we want to automate. Then, if there is any process to be automated, we highlight the intentional elements that are related to these processes (goals and tasks in the *i** model). Those elements will be related to the information and/or entities to be implemented by the intended system. From the list of identified intentional elements we obtain an initial skeleton of OO-Method conceptual model through the application of a set of transformation guidelines (second sub-process, see Fig.1).

Table 1 depicts a summary of the transformation guidelines that are used to explain the problems presented in this paper, which is a subset of the guidelines presented in [2]. This table shows the *i** constructs involved in the transformation, the additional

[1] CAPES-DGU: Integration of Organizational Modelling Techniques to Software Automatic Generation: OO-Method Case (in Portuguese). 2nd partial report. Ministério da Educação, Coordenação Geral de Cooperação Internacional Programa Brasil-Espanha da CAPES/DGU. Processo Nº 167/08, Brazil, 2010

information that must be considered to perform the transformation, and the target constructs of the OO-Method class model.

Table 1. Guidelines for the transformation of *i** models into OO-Method class models.

***i** Construct*	**Additional Information**	**Class Model Construct**
Actor		Class
Resource	Physical entity	Class
	Informational entity related to a physical resource or an actor	An attribute that represents information of the class generated from the actor or physical resource
	Resource in a decomposition tree	Input arguments for the service generated from the related task
	Dependum resource	Input argument of the depender task
	Physical entity inside of an actor boundary	An association between the classes generated from the physical resource and the owner actor
Task	Participating in a resource dependency as depender or dependee	A service of the class generated from the dependum resource
	If generates a resource	A creation service of the class generated from the resource
Dependency link	Where the *dependum* resource and the *depender* and *dependee* actors are transformed in classes	Associations are automatically defined among the generated classes

In order to illustrate, we present a brief example i* model (see Fig. 2) that is defined from the OO-Method case study presented in [11], which is related to the operation of a Photography Agency. This case study is also used in [1, 2]. In particular, the presented i* model shows the reception of work requests (i.e. job applications) from photographers that want to be hired. Due to space constraints, only a simplified version of the complete case study is presented. It is important to mention that, in the complete i* model, not all the i* elements are involved in the transformation process. Only those elements that are related to the intended system are considered (i.e. the involved actors).

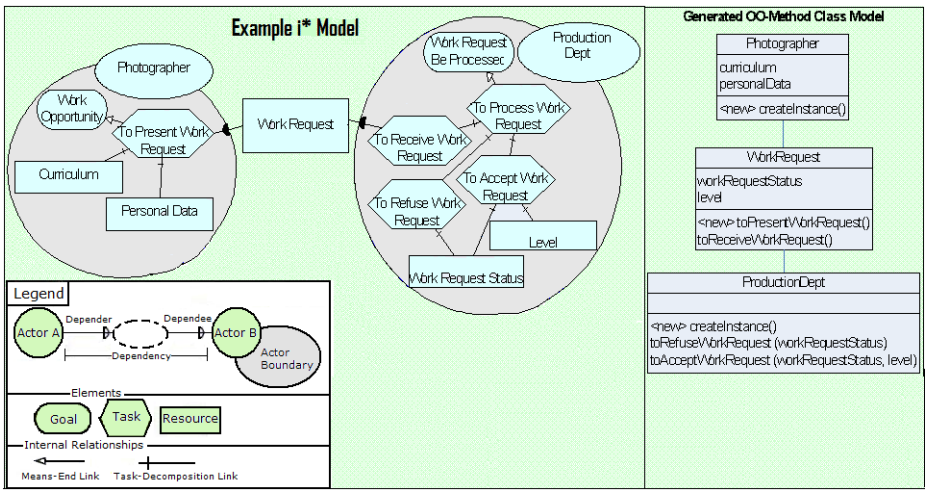

Fig. 2. A illustrate example

3 Some Problems and Solutions

In this section, we show some of the most relevant problems identified to perform an automatic transformation of *i** models into OO-Method Class Diagram, as suggested by the previously guidelines. For each issue a particular solution is proposed.

Problem 1. *It is not possible to automatically infer if a resource corresponds to a physical or an informational entity*. Since a physical entity is transformed into a class and an informational entity is transformed into an attribute, this distinction must be established. As a solution, we propose to extend resources with an attribute which defines the its type because we pretend.

Problem 2. *Differences in the Abstraction levels of i* and OO-Method.* The *i** requirements technique is oriented to capture aspects of the strategies and intentions involved in the relationships among actors (stakeholders), while the OO-Method is concerned with the representation of the functionality of the intended software system. Note that there is some abstraction gap. Furthermore, the transformation guidelines should only consider the subset of *i** elements that are required for the generation of an initial OO-Method class model. However, it is very important to keep the traceability information between *i** and OO-Method models. One possibility is to define an auxiliary model to record the traceability data. This intermediate model could be used specially for those *i** elements that do have direct representation in the OO-Method class model, e.g. goals.

Problem 3. *Two or more kind of elements of the i* model can be transformed into the same kind of element of the OO-Method class model.* As Table 1 shows that both actors and resources may be transformed into classes. Therefore, if we examine only the Class Diagram it is not possible to determine if it has been generated from an i* actor or resource. In other words, the traceability between the conceptual representation of the system and the corresponding requirement element is lost. This problem could also be solved by the intermediate model introduced as solution for the problem 2.

Problem 4. *Some relevant information of the i* model may be lost in the transformation process.* After the application of the transformation guidelines, it is not possible to identify from the generated Class Models: (i) which elements are related to the *depender*, *dependeee*, and *dependum* in the dependency links; (ii) the involved tasks decompositions; (iii) the services that are representing a *means* at the *i** models to preserve the means-end-links. The intermediate model presented as solution for problems 2 and 3 can also store the mapping required to identify these elements from the generated class model.

Problem 5. *It is not possible to directly specify which elements of the i* model must be automated.* According to the proposed transformation process (see Section 2), the transformation guidelines are only applied to those *i** elements that must be automated into the software system. Thus, to capture this information, we propose to use a metamodel extension mechanism to label the corresponding *i** model, for instance, such a UML profile [5]. In addition, the metamodel extension mechanism can also be

used to add the additional properties that are required to automate the transformation guidelines, such as the additional property that is required to solve Problem 1.

Problem 6. *The cardinalities of the associations between classes cannot be automatically inferred.* This problem is due to the difference in the abstraction level of *i** and OO-Method models. As a solution, we propose the introduction of a new property in the *i** model that allows the cardinality of the association among the generated classes to be automatically inferred. In fact in the context of Software Product Line development we have already proposed an i* extension that deals with cardinality (the so called i*-c) [16].

4 Conclusions and Further Work

In this paper we outline our attempt to relate intentional information described in terms of i* models and OO-Method conceptual models. Moreover, we highlight some shortfalls and discuss possible solutions for some of the identified problems.

Our proposal defines guidelines which be automated as well as some procedures which are semi-automatic or even manual, i.e. require human intervention [2]. The solutions presented in this paper are oriented towards the fully automation of the process. Thus, we want to minimize the dependency on highly experienced analysts and designers to manually transform the requirements models into appropriate OO-Method models.

Initial results of our approach are presented in [6]. However, it is important to note that the quality of the GORE (i*) models directly affects the quality of OO-Method conceptual models. In our proposal, we assume that the *i** models are of high standard, i.e. do not present defects (omissions, inconsistency, erroneous facts, ambiguous, etc.). However, this assumption may be unrealistic. Thus, we are also working in proposal to evaluate the quality of requirements models [4, 15].

As future work, we plan to apply the transformation guidelines to different case studies in order to evaluate the correctness and completeness of our proposal. In addition, we plan to formalize and automate the guidelines using metamodeling standards (such as MOF [12]) and model-to-model transformations technologies (such as ATL [8]). Finally, we also consider the definition of metamodel extensions for the *i** framework in order to improve the modeling facilities for MDD environments and to completely automate the transformation of GORE models since we intend to preserve the automate trace between rationales and the data design.

References

1. Alencar, F., Pastor, O., Marín, B., Giachetti, G., Castro, J.: Aligning Goal-Oriented Requirements Engineering and Model-Driven Development. Poster in the 11th International Conference on Enterprise Information Systems (ICEIS´09), May, Milan, Italy (2009)
2. Alencar, F., Pastor, O., Marín, B., Giachetti, G., Castro, J., Pimentel, J.: From i* Requirements Models to Conceptual Models of a Model Driven Development Process. 2nd W. Conf. on the Practice of Enterprise Modeling (POEM´09), Stockholm, Sweden (2009)
3. Care Technologies Company. Available at www.care-t.com. Last access: March (2010).

4. Franch, X.: A Method for the Definition of Metrics over i* Models. In: 21st Int. Conf. on Advanced Information Systems (CAiSE 2009), pp. 201--215a. Springer-Verlag LNCS (2009)
5. Giachetti, G., Marin, B., Pastor, O.: Integration of Domain-Specific Modeling Languages and UML through UML Profile Extension Mechanism International Journal of Computer Science and Applications, vol. 6 nº 5, 145--174 (2009)
6. Giachetti ,G., Alencar, F., Marín, B., Pastor, O., Castro J.: Beyond Requirements: An Approach to Integrate i* and Model-Driven Development. In: XIII Ibero-American Conference on Software Engineering (CIbSE2010), April, Cuenca, Ecuador (2010)
7. Grau, G., Franch, X., Ávila, S.: J-PRiM: A Java Tool for a Process Reengineering i* Methodology. In: RE 2006: p.352--353 (2006)
8. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming, vol. 72 nº 1-2, 31--39 (2008)
9. Lamsweerde, A.v.: Systematic Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley, (2008).
10. Lucena, M., Santos, E.,Silva, M. J. , Silva, C., Alencar, F. , Castro, J.: Towards a Unified Metamodel for i*. In: 2nd IEEE Int. Conference on Research Challenges in Information Science (RCIS'08), Marrakech. Proceedings of the RCIS'08, pp. 237--246 (2008)
11. Marín, B., Giachetti, G., Pastor, O.: The Photography Agency: A case study of the OO-Method Approach. Technical Report DSIC-II/13/08, Universidad Politécnica de Valencia, Valencia, España (2008)
12. OMG: MOF 2.0 Core Specification (2006)
13. OMG: Business Process Modeling Notation version 1.1 (2008)
14. Pastor, O. and Molina, J. C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, Springer-Verlag 1st ed., Springer, New York, New York (2007)
15. Ramos, R.A.: AIRDoc - An Approach to Improve the Quality of Requirements Documents: Dealing with Use Case Models. PhD Thesis. Federal University of Pernambuco, (2009)
16. Silva, C., Borba, C., Castro, J.: G2SPL: A Goal Oriented Requirements Engineering Process for Software Product Line (In Portuguese: G2SPL: Um Processo de Engenharia de Requisitos Orientada a Objetivos para Linhas de Produtos de Software). In: Proceedings of 13th Workshop on Requirements Engineering (WER'10) (2010)
17. Yu, E.: Modelling Strategic Relationships for Process Reengineering, PhD Thesis, University of Toronto, Toronto, Canada (1995).