ALIGNING GOAL-ORIENTED REQUIREMENTS ENGINEERING AND MODEL-DRIVEN DEVELOPMENT

Fernanda Alencar[1,2], Oscar Pastor[1], Beatriz Marín[1], Giovanni Giachetti[1]
[1]*Technical University of Valencia, Avenida de los Naranjos s/n, Valencia, Spain*

Jaelson Castro[2]
[2]*Universidade Federal de Pernambuco, Centro de Informática, Av. Prof. Luiz Freire S/N, 50740-540, Recife, Brazil*

Keywords: Goal-orientation, Requirements engineering, Requirements modelling, Conceptual modelling, MDD.

Abstract: In order to fully capture the various system facets, a model should have not only software specifications but it should integrate multiple complementary views. Model-Driven Development (MDD) and Goal-Oriented Requirements Engineering (GORE) are two modern approaches that deal with models and that can complement each other. We want to demonstrate that a sound software production process can start with a GORE-based requirements model and can finish with advanced MDD-based techniques to generate a software product. Therefore, we intend to show that GORE and MDD can be successfully put together.

1 INTRODUCTION

Recently, the focus on software development is moving from the simple code writing to the models specification and models transformations to obtain the code. A model should have not only software specifications, but also multiple complementary views: intentional, structural, responsibility, functional, and behavioral.

Model-Driven Development (MDD) and Goal-Oriented Requirements Engineering (GORE) are two modern approaches that are generating a lot of literature. Nevertheless, few rigorous attempts to show how they can be properly being joined exist.

Model-driven development methods were devised to take advantage of using models. MDD defends that quality software must be seen as a sequence of well-defined model transformations (Mellor, 2003).

Goal-Oriented Requirements Engineering (van Lamsweerde, 2008) focuses on the activities that precede the formulation of software system requirements. It is said that GORE is an adequate approach to dealing with more and more complex software systems. However, too often works on GORE stay at the Requirements Engineering level, and how to go from their requirements models to the corresponding software products remains basically undefined.

In order to fill up this gap, there are researches focusing on mechanisms that facilitate the generation of a software system from early requirements specifications. For instances, (van Lamsweerde, 2008) considers the KAOS method, (Martinez, 2008) presents an approach inserted at the Tropos context (Giorgini, 2005) including the i* framework (Yu, 1995), and (Alencar, 2003; Santander, 2002; Castro, 2001) propose transformations from i* models to generate UML conceptual models. However these approaches do not use a MDD approach, avoiding the possibility to automatically generate the final software product.

The main contribution of our work is mainly practical: we show that GORE and MDD can be successfully combined. On the GORE side we select the original i* framework. From the MDD part we have taken the OO-Method approach (Pastor, 2007) and its industrial tool - Olivanova (Care, 2008) - that generates a complete software product through a compilation process from an OO Conceptual Model. The last piece of that puzzle is to select a real problem: we choose a domain based on an Agency of Photographers that was solved in the PROS Research Center. From the requirements description, we construct an early requirements model to represent the organizational context. Next, we will deal with late requirements and we will introduce the information system that will automate some services. Then we compare this model with the conceptual schema (an extended class diagram) previously created with OO-Method. So that, we will answer the question: "Can we use the i* framework as the requirement model for OO-Method?" Additionally, we can evaluate how much of the involved process can be automated.

To achieve these objectives, section 2 presents the modeling process used to obtain the initial evidence for using the two approaches together (i* framework and the OO-Method MDD approach). Finally, the section 3 summarizes the paper and points out future works.

2 THE APPLICATION OF THE I*

It seems interesting that the requirements models can be more expressive and capture intentionality, rationales, and alternatives; but above all, that it will possible to transform the i* requirements model into the conceptual model of the OO-Method MDD approach. We use a pre-existing industrial case study: the Photographic Agency, which was modelled using the OO-Method approach. For reasons of space, we show a partial view. A complete version can be found in (PROS, 2008).

2.1 The Research Methodology

In order to answer our main question: *Can we use the i* framework as the requirement model for the OO-Method approach*?, we have structured our research in five steps: (1) elicitation of basic requirements and construction of initial description in natural language, (2) model the business processes using the i* models (the Strategic Dependency model – SD- and the Strategic Rationale model - SR), (3) seek to automate some business process by the insertion of a information system, (4) comparison of the conceptual models get in the OO-Method context with the i* models lately drew in order to identify the common elements, and (5) definition of a subset of i* elements that can be used as an intermediate model, or as a requirement model in the OO-Method approach.

2.2 Applying the Metodology

In the first step, we construct a requirements description of our case study, which deals with the business processes of making available the photographic report by a publishing house. The photography agency manages the photo reports and their distribution for publishing houses that operate with freelance photographers. These freelances present requests to the photography agency. The publishing houses also must be subscribed into the agency. Usually, the publishing houses are attended by the production department, which searches in the file of photo reports the more interesting reports according to the topics asked by the publishing houses. If a publishing house request for a photo report, this is sent with a delivery note that must be signed by the publishing house, and brought by messengers.

In the second step, all the actors involved are identified: the Publishing House, the Production Department (Dep.) and the Messenger (Figure 1). After that we determine the intentional dependency among the actors. For example, the Publishing House need that the Production Dep. satisfies its goal of *Some photo report be make available* (a goal dependency). Moreover, the *Publishing House* needs that the *Production Dep.* makes available a *list with the description of the different photo reports* (a resource dependency) that it has. In addition, the *Production Dep.* requires that the *Publishing House* make available the *Desired Topics* and its *Report request* (both are resource dependency). Some other dependencies are specified in the i* model and we have detected that all correspond to resource dependencies. This reflects the viewpoint of the developer whom made the textual description. In particular, since the OO-Method focuses on functional aspects, we explain this fact below.

Afterwards, we began to expand the actors (SR model) to identify the intentions and the "Why´s" of dependency relationships among the actors. The Figure 1 shows a partial view of the SR i* model. In this model, almost all elements are tasks which are broken down into other tasks (task-decomposition link in the SR model). For example, at the *Production Dep.* actor we see only two goals (*The Publishing House be attended* and *The description of different photo reports be sent*), which have been modeled as the ends to be reached by some means (a means-end link in the i* SR model). These intentional goals were inferred from reading the Photography Agency description. They were not explicitly raised.

In the third step of the research methodology, we introduced an actor that represents the system that we need. We call it by *PAS* (Photographic Agency System), which will not be expanded.

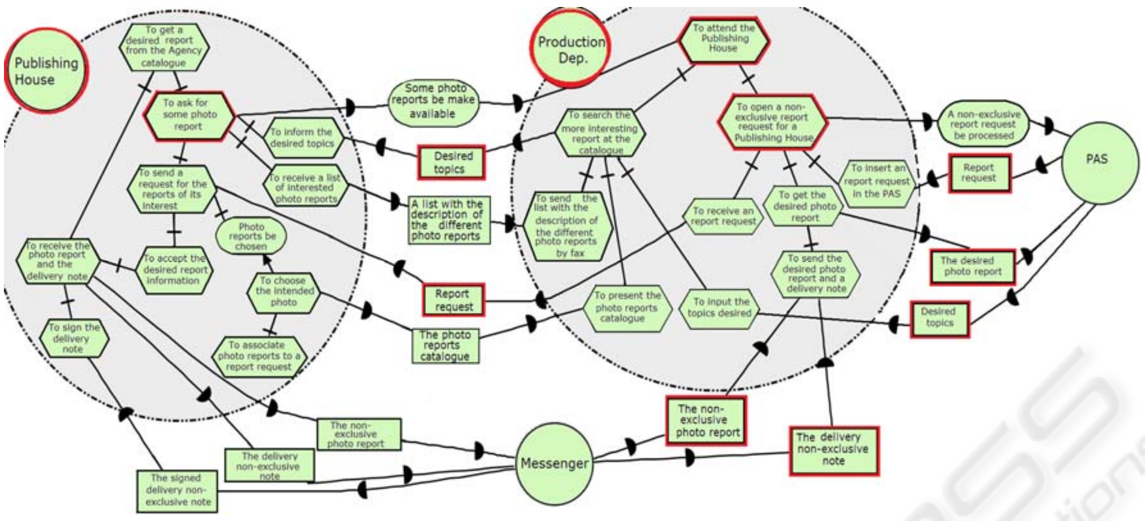
Publishing House
To get a desired report from the Agency catalogue
To ask for some photo report
To inform the desired topics
To receive a list of interested photo reports
To send a request for the reports of its interest
Photo reports be chosen
To receive the photo report and the delivery note
To accept the desired report information
To choose the intended photo
To sign the delivery note
To associate photo reports to a report request
Some photo reports be make available
Desired topics
A list with the description of the different photo reports
Report request
The photo reports catalogue
The non-exclusive photo report
The delivery non-exclusive note
The signed delivery non-exclusive note
Messenger
Production Dep.
To attend the Publishing House
To search the more interesting report at the catalogue
To open a non-exclusive report request for a Publishing House
To send the list with the description of the different photo reports by fax
To receive an report request
To get the desired photo report
To insert an report request in the PAS
To send the desired photo report and a delivery note
To present the photo reports catalogue
To input the topics desired
The non-exclusive photo report
The delivery non-exclusive note
A non-exclusive report request be processed
Report request
The desired photo report
Desired topics
PAS

Figure 1: The Organizational SR model for the Photographic Agency.

By the client decision, only the Agency´s Departments actors will interact with the system (an internal system). The *Production Dep.* will do its work with the help of the *PAS*. Thus, some processes now will be done by the *PAS*. The *PAS* must record information about the *Publishing House*, its *Photo Report* request, its desired Topics, and the *Photo Report*. Some others business processes are not automated and continue to depend on the relationship with the *Publishing House*, the *Production Dep.* and the *Messenger* (whom deliver orders).

In the fourth step, the comparison between the i* model and the OO-Method class diagram has been developed according the problem description. In this diagram the classes correspond to the functional elements of the final system. Due to space constraints, we only present the resultant class diagram in the Figure 2.

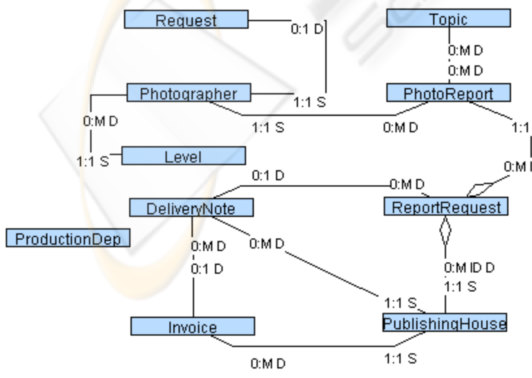

Figure 2: The Conceptual Class Diagram of the Photographic Agency System for the Publishing House Request.

In our comparison between the i* model (Figure 1) and the conceptual class diagram (Figure 2) we only focus on the request for a desired photo report. From the interaction of the *Production Dep*. actor with the *PAS* actor we find a particular goal: *A non-exclusive report request must be processed*. The satisfaction of a goal may include the execution of several tasks, and the production and usage of resources in the i* models. We started our studies highlighting those resources in the i* model (Figure 1). By definition, a resource is an entity, physical or informational. Then, in Figure 2 we found that all these elements were modeled as classes.

Thus, we are appraising that we have three options for modeling, specifically when we have a resource dependency: (1) if the resource element it is not related with the system it will not modeled; (2) if the resource is related with the intended system and represents an entity, then it will be represent by a class; (3) if the resource is related with the intended system and represents an informational entity, then it will be modeled by a class attribute.

We must study the other i* elements. Therefore, we look for the actors and try to answer the questions: *Who are the actors whose data must be captured and store by the system?* and *Who are the actors that we do not store its data but directly use the system?*

For the first question we met the actor *Publishing House*. This element was modeled as a class in the class diagram. For the second question, we met the actors *Production Dep*. using the system. Thus, we highlight this element (Figure 1) and, we observe that this element is an agent class in the class diagram. This distinction is done by the OO-method approach (Pastor, 2007). The actor (*Messenger*) does not use the system and we do not need to store his/her data, or any information related When we are looking for an actor we will have: (1) to represent it as class, as in the first case; (2) to represent it as agent, as in the second case; or (3) do not represent it at the class diagram. When the decomposition of the task *To ask for some photo report* at the *Publishing House* actor boundary is analyzed, we identify that it will be necessary to choose the intended reports and to associate these photos to a report request. Thus, we deduced that will have an association among three classes: *Publishing House*, *Report request*, and the *desired photo repots*. Despite the class name, the Figure 2 models this situation as an *aggregation* relationship.

In spite of the main strengths of the OO-Method approach, which allow the automatic generation of information systems from well-defined conceptual models, this method also presents disadvantages, such as the lack of traceability between the classes (of the class diagram) and the requirements from which these classes are generated. Thus, when someone read the class diagram, some information is not presented. It is true that not all the information captured in an early requirement phase is useful in the implementation of object-oriented systems. However, some of them might to be. In our research, we encounter these evidences by using a rich requirement ontology (like i* framework) that can be very useful for us to go toward building conceptual models richer.

3 CONCLUSIONS

Summarizing, we try to answer the question: *Can we use the i* framework as the requirement model for the OO-Method?*" We conclude that it is possible to use the i* framework. Indeed, we need to derivate an intermediate requirements model with rationales from which, using transformations functions on MDD, derive some elements of the class diagram. Sure, the initial phase needs a more abstract model while the final phases of development need a more concrete model. Thus, some arguments should be introducing in the later phases models. Therefore, we can reduce the gap from requirements specification in the problem space to conceptual models in the solution space. In sequence, we will formalize this proposing a simple and direct way of extract from a requirement models the main elements that the OO-Method conceptual model needs.

At this moment we are working to formalize concrete guidelines to determine an intermediate requirement model from which we can generate a conceptual model as the model of the OO-Method approach. To perform this generation, we are focusing on model transformations that are based on the metamodels of the involved models.

REFERENCES

Alencar, F., Pedroza, F., Castro, J., Amorim, R., 2003. New Mechanism for the Integration of Organizational Requirements and Object Oriented Modeling. In Proc. of the VI Workshop on Requirements Engineering (WER 2003), Piracicaba, Brazil. November, pp.109-123

Care Technologies Company, 2008. OlivaNova Suite. Availble at www.care-t.com.

Castro, J., Alencar, F., Cysneiro Filho, G. , Mylopoulos, J., 2001. Integrating Organizational Requirements and Object Oriented Modeling In: 5th IEEE International Symposium on Requirements Engineering, 2001, Toronto. Proc. of RE´01. IEEE Press, 2001c. p.146 – 153.

Giorgini P., Mylopoulos J.,Sebastiani R., 2005. Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology. In Engineering Applications of Artificial Intelligence, Elsevier 18(2), March.

I-star wiki, 2008. i* Guide. Available in http://istar.rwth-aachen.de/tiki-view_articles.php.

Martínez, A.: Conceptual Schemas Generation from Organizational Models in an Automatic Software Production Process, PhD Thesis, Polytechnic University of Valencia, Spain (2008)

Miller, J., Mukerji, J., 2003. MDA Guide Version 1.0.1.

Mellor, S. J., Clark, A. N., Futagami ,T., 2003. Guest Editors" Introduction: Model-Driven Development". IEEE Software, vol. 20, no. 5, Sep/Oct, pp. 14-18.

Pastor, O. and Molina, J. C. Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, Springer-Verlag New York, USA (2007)

PROS: The Photography Agency: A Case Study of the OO-Method Approach. Available at: http://www.pros.upv.es/. Last access: Dec (2008).

Santander, V., Castro, J.: Deriving Use Cases from Organizational Modeling. 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002), Essen, Germany. September, pp. 32-42 (2002)

van Lamsweerde, A., 2008. Requirements Engineering: From Craft to Discipline. ACM SIGSOFT 2008/FSE-16, November 9–15, Atlanta, Georgia, USA.

Yu, E. et al., Strategic Actors Modeling with i*. Tutorial Notes. 2008. International Conference on Requirements Engineering (RE'08), IEEE Computer Society, Spain, pp. 01–60.