

A model-driven framework to integrate Communication Analysis and OO-Method: Supporting the model transformation^{*}

Marcela Ruiz¹, Sergio España¹, Óscar Pastor¹, Arturo González²

¹ Centro de Investigación en Métodos de Producción de Software PROS, Universitat Politècnica de València

{lruiz, sergio.espana, opastor}@pros.upv.es

² Departamento de Sistemas Informáticos y Computación DSIC, Universitat Politècnica de València

agdelrio@dsic.upv.es

Abstract. Models are now part of an increasing number of engineering processes. However, in most cases, they are confined to a simple documentation role instead of being actively integrated into the engineering process. The model-driven development approach considers models as first-class entities and also considers tools, repositories, etc. as models. In order to take full advantage of these ideas, model transformation emerges as a main activity. Model transformation aims at supporting the production of target models from a number of source models. Following the model-driven development paradigm, we propose a model-driven framework to integrate Communication Analysis (a communication-oriented business process modelling and requirements method) and OO-Method (an object-oriented model-driven development method). This integration framework is composed of two stages: diagramming support and transformation support. This paper describes the second stage (the first stage was described in previous works). Phases, tasks, technological support and examples are presented. Finally, we conclude with an analysis and discussions about lessons learned and an evaluation proposal to assess the usability of the transformation module.

Keywords: Model transformation, model-driven development, Communication Analysis, OO-Method, requirements engineering, Eclipse Modelling Framework, ATL.

* Research supported by the Spanish Ministry of Science and Innovation project PROS-Req (TIN2010-19130-C02-02), the Generalitat Valenciana project ORCA (PROMETEO/2009/015), Santiago Grisolia grant (GRISOLIA/2011/005) and co-financed by ERDF structural funds.

1 Introduction

Information system (IS) development demands requirements methods to establish organisational needs. The model-driven development (MDD) paradigm offers some advantages to requirements models such as the potential to derive conceptual models for automatic software generation. Although there has been a successful evolution of software projects, the last CHAOS report shows that 68% of projects have failed or threatened [1]. Academy and industry have agreed to designate the lack of user participation to be a risk factor that threatens software projects [2]. Involving the user in the development process allows the early correction of mistakes and increases the acceptance of the final software product [3]. An effective solution for involving the user in the software development process is to provide requirements engineering practices; however, there are still some difficulties in the industrial application of requirements engineering methods [4].

The MDD paradigm has the potential to overcome some of the difficulties related to the industrial adoption of requirements engineering practices. Information system requirements specifications have documented the need for an organisation that provides the necessary support for work practices and communications [5]. Since specifications are essentially models, the MDD paradigm can get the most out of them: For instance, requirements models can be used to derive conceptual models and the management of trace links can be facilitated. This new role of requirements models, which surpasses their current status of documentation to become the main development artefact, increases their industrial value.

However, the statements above still open challenges in model-driven requirements engineering [6]. Since the requirements engineering methods that are available in the literature are not fully compatible with MDD environments, we have developed a generic model-driven framework to adapt existing requirements engineering methods into the MDD paradigm (see a fully detailed description of the framework in [7]). As a practical application of the integration framework, we target a specific requirements engineering method called Communication Analysis and a MDD development environment called OO-Method.

Communication Analysis is a communication-oriented business process modelling and requirements method that proposes the analysis of information systems from a communicational perspective [8]. This method is currently applied in complex projects in industrial environments. Due to the fact that this method is used in practice, the integration of Communication Analysis into a MDD framework poses interesting and practical research challenge: to offer technological support for modelling the requirements engineering models and to facilitate conceptual model derivation and model transformation. By providing a proper technological platform for the requirements method, the adoption of the method by large and medium-size enterprises can be expected to increase. Also, the integration of this method into the MDD environment will allow the derivation of the conceptual models as well as the automatic generation of the final software product. Indeed, we have chosen the OO-Method (an

object-oriented software development method) because it is a MDD framework that has industrial tool support in Integranova¹.

The integration framework suggests two stages: The Stage 1 aims at providing diagramming support and is related to defining the requirements method metamodel and designing a modelling tool. This stage has been presented in previous works (for more details on this stage, please see [9]). This paper focuses on the Stage 2, which aims at providing support for model transformation. The main contributions are the following:

- The Stage 2 of the integration framework is presented in full detail. Results and lessons learned are discussed indeed.
- There is significant development of a framework to support method integration.
- Evaluation proposal is provided to assess the usability, agility, and ease of use the transformation module instead of following manual techniques.

The paper is structured as follows: Section 2 reviews the related works. Section 3 presents the model-driven framework for integrating Communication Analysis and the OO-Method. Section 4 describes Stage 2 of the integration framework (phases, tasks and results) in detail. Section 5 discusses the results, some lessons learned, and a validation proposal to assess usability, agility, and ease of use. Section 6 presents our conclusions and future lines of work.

2 Related works

The diversity of software development and changing in software applications has given rise to a need to combine and integrate different methods [10]. There are several works on method integration: the integration and combination of several methods or the parallel execution of methods [11]. Following, we review existing method integration frameworks, we discuss the current situation of model-driven requirements engineering, and we comment on state-of-the-art technological support for business process modelling notations.

A metamodel proposal was presented by Saeki [12], which proposes an approach to integrate multiple methods through the use of metamodels. The approach is based on two aspects: the use of semantic equivalence between method components to establish uniformity between individual methods and the incorporation of procedural information, tasks, and their order in the metamodel. We have used the experience presented in this proposal to specify the abstract representation of the Communication Analysis and OO-Method. The CASE tool presented in this proposal only supports the metamodel specified in it (Object diagrams, state transition diagrams, and data flow diagrams). In addition, we provide a modelling tool to support Communication Analysis requirements models that is also open-source and can interact with other modelling environments developed in Eclipse.

¹ CARE Technologies – Integranova Model Execution System <http://www.integranova.es/>

An interoperability proposal is presented by Giachetti 2011 [13], which proposes an approach to achieve interoperability in MDD processes. This interoperability approach is based on current metamodelling standards, modelling language customization mechanisms, and model-to-model transformation technologies. The proposal presents the integration of modelling languages to obtain a suitable interchange of modelling information and to perform automatic interoperability verification. This proposal integrates several modelling languages taking into account a MDD perspective, i.e., model-to-model transformation technologies to obtain interoperability among different method perspectives (object-oriented, goal-oriented, communicative-oriented, etc.). This proposal uses the metamodel concept proposed by Saeki and also includes MDD practices and mechanisms to obtain benefits such as suitable technologies, modelling language customization, and model-to-model transformation.

A complete solution that includes requirements models as part of MDD processes is still not available [6]: There is little tool support to manage requirements models, and to automate model transformation and code generation activities and there is little use of models to describe requirements in the MDD context. These are some of the challenges that still exist in the MDD community. The proposal by de la Vara 2011 [14] presents a methodological approach for business process-based requirements specification and object-oriented conceptual modelling of information systems. The approach consists of four stages: organisational modelling, purpose analysis, specification of system requirements, and derivation of object-oriented diagrams. This proposal integrates system requirements and OO conceptual modelling. The main idea is to determine the correspondence between the system requirements and the IS. This proposal is oriented to solving some problems detailed by Loniewski in [6], but tool support and automation support are not provided by this proposal.

Currently there are different tools for supporting business process modelling: AuraPortal BPM², MOSKitt³, and Microsoft Visio⁴. However, these tools do not provide support for Communication Analysis requirements model, and are not open-source, which reduces their approval and use by the academic community.

3 A model-driven framework to integrate Communication Analysis and OO-Method

As a first step, we have conceived a general framework that defines stages, phases, and tasks that aim at integrating requirements methods in MDD environments. This framework takes into account modelling activities and model transformation activities (for more details of this framework please see [7]).

Our goal is to integrate Communication Analysis and OO-Method. Therefore, we have put into practice the proposed general framework (See Fig. 1).

² AuraPortal BPMS. Available: <http://www.auraportal.com>

³ MOSKitt. Modelling Software Kitt. Available: <http://www.moskitt.org/>

⁴ I. T. Corporation. Visio Stencil and Template for UML 2.2. Available: <http://softwarestencils.com/uml/index.html>

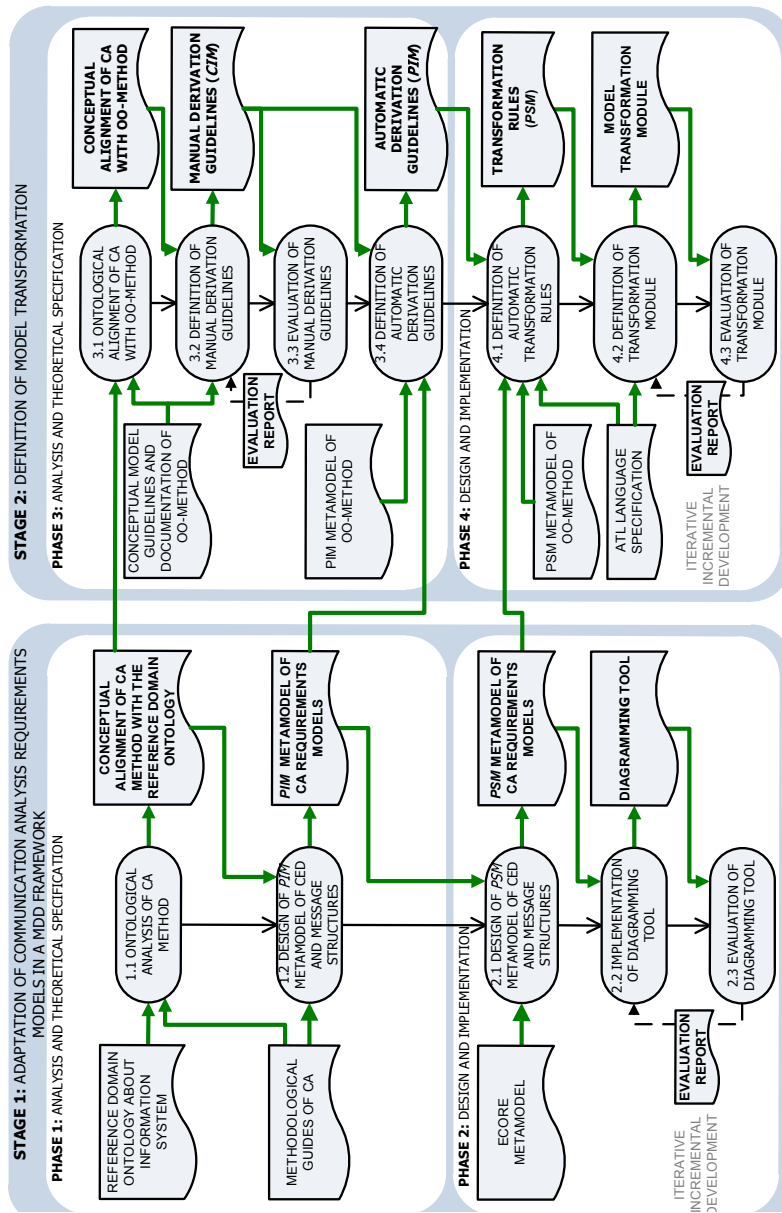


Fig. 1. Model-driven framework to integrate Communication Analysis and OO-Method

Each stage of the framework is divided into two phases, and each phase has several tasks. We have decided to differentiate each stage according to the tasks involved in the MDD process. We distinguish between the stage related to modelling tasks (Stage 1) and the stage related to model transformation tasks (Stage 2).

Model-Driven Architecture (MDA) is an approach that uses models in software development. MDA proposes the well-known and long-established idea of separating the specification from the system operation [15]. The application of this concept for developing frameworks to integrate methods allows us to build artefacts with different levels of abstraction. This means that artefacts are available at different abstraction levels according to the MDA layers. The MDA layers are: computation-independent models (CIM), platform-independent models (PIM), and platform-specific models (PSM).

Since a metamodel is a model, the MDA approach can be applied to support requirements engineering methods, i.e., we differentiate the abstract representation of the Communication Analysis requirements models according to MDA layers. For this reason, we distinguish between the PIM metamodel and the PSM metamodel. Having the metamodel at different levels of abstraction allows us to implement the PIM metamodel in different target platforms according to the technological platform chosen.

In the same way, we apply the MDA architecture to build the artefacts corresponding to the Stage 2. The derivation guidelines are specified in CIM and PIM layers to allow different technological platforms.

Stage 1 involves some activities that are related to metamodel specifications and construction of diagramming tools. Four products have been obtained at this stage: The first is the conceptual alignment of Communication Analysis with the reference ontology of the information system. This alignment allows the principal concepts and primitives of the Communication Analysis method to be distinguished. Some examples of conceptual frameworks and ontological analyses are available in [16]. The second is the PIM metamodel specification of Communication Analysis requirements models. This metamodel contains a set of elements (metaclasses) and relationships (associations) that represent the concepts of the method. Each metaclass and association corresponds to a concept of the ontology. The metamodel is at a PIM level because it does not have technological information (i.e., the target platform has not yet been considered). The third is the PSM metamodel specification of Communication Analysis requirements models. This metamodel specifies metaclasses and associations with platform-oriented information. The fourth is the diagramming tool. This supports the modelling activities of the Communication Analysis requirements models (communicative event diagrams and message structures). For more details about this stage, please see [7, 9].

Stage 2 involves some activities related to model transformation from requirements models to conceptual models. These activities are aimed at the generation of software code in an automatic way. This stage is presented below.

4 Definition of Model transformation

The task carried out in Stage 1 (Phase 1 and Phase 2) allows us to obtain a modelling environment to specify Communication Analysis requirements models (communicative event diagrams and message structures). By following a set of derivation

guidelines [17], it is possible to obtain conceptual models from Communication Analysis requirements models.

As presented in section 3, a set of tasks can be followed to build a transformation module. This module should support the transformation rules.

Fig. 1 presents the phases and tasks corresponding to Stage 1 and Stage 2. This paper focuses on Stage 2 in order to analyse each phase and task. Stage 2 aims at defining model transformations from requirements models to conceptual models in a MDD environment. Each task corresponds with MDA layers [15].

Stage 2 has two phases (Phase 3 and Phase 4): The goal of Phase 3 is to reason about the derivation guidelines. In order to provide derivation guidelines, the method engineer (the participant role of this task) uses the conceptual alignment of Communication Analysis concepts obtained in Phase 1 (please see Fig. 1) to establish the correspondences among the concepts of two methods being integrated. Next, the method engineer designs a set of manual derivations according to the correspondences established. Finally, the method engineer specifies the manual derivation guidelines in a pseudo-code for their future implementation. Phase 3 has four tasks: 3.1 Ontological alignment of Communication Analysis with OO-Method conceptual models; 3.2 Definition of manual derivation from Communication Analysis to OO-Method; 3.3 Evaluation of derivation guidelines; and 3.4 Definition of automatic derivation guidelines. These tasks are explained in Table 1. Participant roles are presented. One person can to play several roles.

Table 1. Tasks corresponding to Phase 3 of the integration framework

TASKS	ENTRIES	OUTPUTS	PARTICIPANT ROLES
3.1 Ontological alignment of Communication Analysis with OO-Method conceptual models.	<ul style="list-style-type: none"> - Conceptual alignment of Communication Analysis concepts with reference domain ontology. - Conceptual model guidelines and documentation of OO-Method. 	<ul style="list-style-type: none"> - Conceptual alignment of Communication Analysis concepts with OO-Method. 	<ul style="list-style-type: none"> - Ontology expert - Method engineer.
3.2 Definition of manual derivation guidelines. From Communication Analysis to OO-Method.	<ul style="list-style-type: none"> - Conceptual model guidelines and documentation of OO-Method. - Conceptual alignment of Communication Analysis concepts with OO-Method. 	<ul style="list-style-type: none"> - Manual derivation guidelines (CIM). From Communication Analysis to OO-Method. 	<ul style="list-style-type: none"> - Method engineer. - Representative analyst
3.3 Definition of manual derivation	<ul style="list-style-type: none"> - Manual derivation guidelines (CIM). 	<ul style="list-style-type: none"> - Manual derivation guidelines 	<ul style="list-style-type: none"> - Representative analyst.

guidelines. From Communication Analysis to OO-Method.	From Communication Analysis to OO-Method.	(CIM). From Communication Analysis to OO-Method.	- Researcher.
3.4 Definition of automatic derivation guidelines.	-PIM metamodel of Communication Analysis. - PIM metamodel of OO-Method. - Manual derivation guidelines (PIM). From Communication Analysis to OO-Method.	- Automatic derivation guidelines (PIM).	- Method engineer. - Representative analyst

The goal of Task 3.1 is to align the concepts of the two methods. Thus, it is necessary to analyse the concepts of the conceptual models and the concepts of the requirements models. The method engineer has to be an expert in Communication Analysis and OO-Method. If the method engineer is only expert in one method, then the participation of more than one expert must be obtained. The ontological alignment is the step previous to designing the derivation guidelines. For more details about this task, please see [18].

The goal of Task 3.2 is to establish a set of derivation guidelines in natural language. This guide should offer steps to obtain the conceptual models manually. The conceptual alignment between Communication Analysis and OO-Method is very important because it allows the correspondences among elements of each method to be established. Examples of the application of the manual derivation guidelines are available in [19-20]. At this point, the method engineer has to be an expert in MDD methods. Fig. 2 presents an example of the derivation of a new class. Rule steps, notes, and pattern matching are provided in order to facilitate their reading and understanding.

The goal of Task 3.3 is to evaluate the manual derivation guidelines to improve them and to assess the quality of the derived conceptual models. This task was carried out in the context of a controlled experiment. One set of subjects applied a text-based conceptual model derivation. Another set of subjects applied the derivation guidelines. The results were compared and interesting results were obtained: a significant impact on conceptual model completeness was shown when the derivation guidelines were applied. The results and feedback of the subjects were part of the evaluation report for improving the derivation guidelines. In addition, several lab-demos were carried out to analyse the use of the derivation guidelines.

The goal of Task 3.4 is to represent the manual derivation guidelines as an algorithm. A pseudo-code of the derivation guidelines is obtained after carrying out this task. The PIM metamodel of the two methods were used to define the derivation guidelines.

Three products were obtained of Phase 3: The first product is the conceptual alignment of Communication Analysis and OO-Method. This product is the first step

in reasoning about the concepts of each method. The second product is the manual derivation guidelines to derive conceptual models from Communication Analysis requirements models. This product presents the derivation guidelines in natural language for human readers. This product was used by students with satisfactory results. It is important to highlight the MDA layer of this product. This product is in a CIM layer of MDA because technological support is not the objective. The third product is the automatic derivation guidelines. This product was developed to specify the derivation guidelines in a computational language, but disregarding the technological platform. For this reason, this product is at a PIM level in the MDA layers.

Rule OM4. Derivation of a new class from an aggregation substructure	
Pattern matching	
1	Each aggregation substructure within the message structure of the event being processed that <i>is not</i> an extension aggregation substructure.
Preconditions	
1	Rule OM2 has already been applied to the requirements model.
Rule steps	
1	Create a new class.
2	Assign to the name of the class the name of its corresponding aggregation substructure, replacing spaces with underscores and using uppercase letters. In any case, the analyst can choose to give a different name to a class.
Traceability information	
1	Create a trace link of type <i>Class_Derivation</i> between the communicative event and the newly-created class (this can be done after step 2).
2	Create a trace link of type <i>Class_Derivation</i> between the aggregation substructure and the newly-created class (this can be done after step 2).
3	If the aggregation substructure describes an organisational role, then create a trace link of type <i>Agent</i> between the organisational role and the newly-created class (this can be done after step 2).
Notes	
1	In case the analyst is creating the conceptual model using <i>Integranova Modeler</i> , then we advice not using <i>extended creation</i> facility optionally offered by the tool (which automatically creates an autonumeric identifier attribute, a creation service, a destruction service, and an editing service). In case this facility is used, the analyst should be aware of the model elements automatically added by the tool and remove those that are not needed.

Fig. 2. Example of manual derivation guide

The goal of the Phase 4 is develop a model transformation tool to carry out model transformation activities among Communication Analysis requirements models and OO-Method conceptual models. In order to provide a transformation module, the analyst uses the PSM metamodel of the requirements models and conceptual models to define the transformation rules. Finally, the analyst should design software evaluation activities for the transformation module. These activities should involve the ex-

perts and the final users. Phase 4 has three tasks: 4.1 Definition of automatic transformation rules; 4.2 Definition of transformation module; and 4.3 Evaluation of transformation module. These tasks are explained in Table 2.

Table 2. Tasks corresponding to Phase 4 of the integration framework

TASKS	ENTRIES	OUTPUTS	PARTICIPANT ROLES
4.1 Definition of automatic transformation rules.	<ul style="list-style-type: none"> - PSM metamodel of Communication Analysis requirements models. - PSM metamodel of OO-Method. - ATL language specification. - Automatic derivation guidelines (PIM). 	<ul style="list-style-type: none"> - Transformation rules (PSM) from Communication Analysis models to conceptual models. 	<ul style="list-style-type: none"> - Method engineer.
4.2 Definition of transformation module.	<ul style="list-style-type: none"> - ATL language specification. - Transformation rules (PSM) from Communication Analysis models to conceptual models. 	<ul style="list-style-type: none"> - Model transformation module. 	<ul style="list-style-type: none"> - Analyst. - Developer.
4.3 Evaluation of transformation module.	<ul style="list-style-type: none"> - Model transformation module. 	<ul style="list-style-type: none"> - Model transformation module. 	<ul style="list-style-type: none"> - Representative analyst. - Researcher

The goal of Task 4.1 is to define the automatic transformation rules in the specific technological platform ATL⁵. Thus, a study about a transformation engine was carried out. QVT⁶ is a standard proposed by the OMG for model transformation that is implemented in the Eclipse⁷ environment. ATL is another transformation engine developed in the GMT⁸ project of Eclipse. The result of this study allows us to choose ATL as the better option to implement the derivation guidelines. The derivation guidelines are expressed in declarative and imperative sentences, so the hybrid nature of ATL provides patterns and a language to specify kind of heuristics of this kind.

Fig. 3 presents an example of a transformation rule specified in ATL language. The PSM metamodel of the two methods being integrated are used as input to specify the

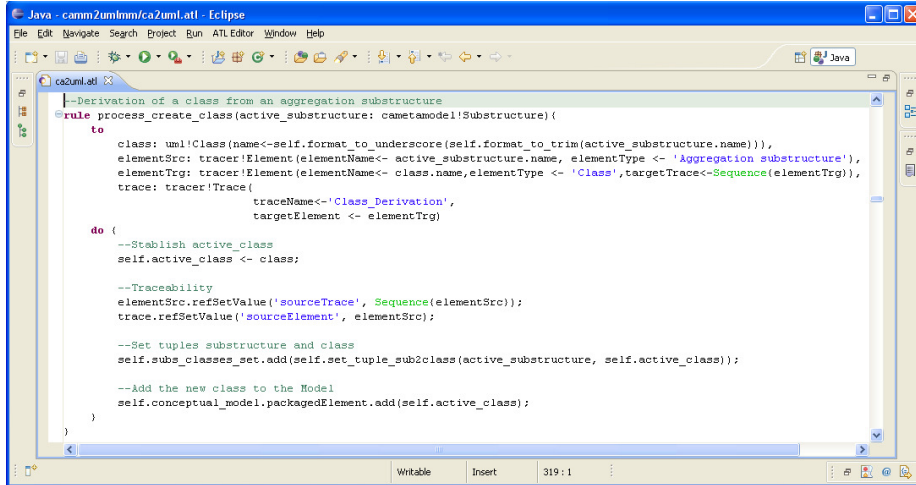
⁵ ATL, the Atlas Transformation Language. Available: <http://www.eclipse.org/atl>

⁶ M2M/QVT, Declarative (QVTd) Available: http://wiki.eclipse.org/M2M/Relational_QVT_Language_%28QVTR%29

⁷ Eclipse.org. Available: <http://www.eclipse.org>

⁸ GMT Project. Available: <http://www.eclipse.org/gmt>

derivation guidelines in ATL language. The PSM metamodels are specified in ECORE language [21].



```

--Derivation of a class from an aggregation substructure
rule process_create_class(active_substructure: cametamodel!Substructure)
to
class: uml!Class(name<-self.format_to_underscore(self.format_to_trim(active_substructure.name)),
elementSrc: tracer!Element(elementName<- active_substructure.name, elementType <- 'Aggregation substructure'),
elementTrg: tracer!Element(elementName<- class.name,elementType <- 'Class',targetTrace<-Sequence(elementTrg)),
trace: tracer!Trace(
    traceName<-'Class_Derivation',
    targetElement <- elementTrg)
do {
--Establish active_class
self.active_class <- class;
--Traceability
elementSrc.refSetValue('sourceTrace', Sequence(elementSrc));
trace.refSetValue('sourceElement', elementSrc);
--Set tuples substructure and class
self.subs_classes_set.add(self.set_tuple_sub2class(active_substructure, self.active_class));
--Add the new class to the Model
self.conceptual_model.packagedElement.add(self.active_class);
}

```

Fig. 3. Example of ATL code: Derivation of a new class from an aggregation substructure

The goal of Task 4.2 is to define a model transformation module (an environment to carry out the model transformation activities). The environment has to support the metamodel of both methods. In addition, it is important to provide the correct format for the metamodel (i.e., Ecore format). The environment has to support the model to be transformed and to show the transformation results to the user. Also, the transformation module has to show messages to the user if something is wrong with the models for future modification.

The goal of Task 4.3 is to evaluate the transformation module to determine whether or not it is usable. This task is part of the future work discussed in section 5.

Two products are obtained from Phase 4: The first product is the transformation rules. This product is specified at a PSM level in the MDA layers. ATL language has been used to specify the transformation rules. The second product is the model transformation module. This module has the transformation environment to support the transformation rules.

5 Discussion

The technological implementation of the derivation guidelines provides the advantage of introducing models into the software development process. However, the model transformation needs to be validated with regard to its performance. We have carried out experiments to evaluate the manual derivation guidelines, but the evaluation of the implemented transformation module is part of our future work. Previous experiments have been carried out with master students to evaluate the quality (i.e.,

completeness and validity) of the generated conceptual models when applying the manual derivation guidelines. In the future, we intend to assess the usability of the transformation module as well as the quality of the resulting models in order to compare them with the outcome of the manual derivation guidelines.

Regarding the model-driven framework presented in this paper, we are aware that it is convenient to apply it to solve other cases. This way, we will be able to assess the scalability of the framework. We anticipate some limitations concerning to the size of the case to solve (For instance, the methods to integrate could be bigger or smaller than the case about Communication Analysis and OO-Method). A MDD tool that supports completely the framework is missing, for this reason the application of cases bigger than the case presented in this research work open a new investigation line to evaluate the scalability of the proposal. An evaluation of the proposed framework is considered as a future work of this research.

6 Conclusions and future work

We were motivated about the possibility of integrating different requirements engineering methods in MDD environments. For this reason, we have proposed a generic framework to do this. The framework itself is model-driven since it differentiates the treatment of method artefacts at the different MDA layers. The framework defines two stages: Stage 1 is aimed at providing support for modelling according to MDD principles (e.g. metamodel conformance), whereas Stage 2 is aimed at providing support to model transformations. In practice, we have applied the framework, to integrate Communication Analysis and OO-Method. This paper presents the Stage 2 of this integration in full detail. The main goal is to allow for the transformation of Communication Analysis requirements models (mainly specified as communicative event diagrams and message structures) into OO-Method conceptual models.

To perform this integration, we first carried out an ontological alignment of the concepts of Communication Analysis and OO-Method so as to set a sound theoretical foundation for the definition of model transformations. Then, we defined a set of manual derivation guidelines [18-20]. These guidelines belong to the CIM level because at this point no automation is intended. Since a certain degree of automated support is intended, the derivation guidelines were defined as a transformation procedure using a platform-independent pseudo-code. Therefore, this procedure belongs to the PIM level. Then, we defined a set of transformation rules to provide the PSM level. Thus, a study about a transformation engine was carried out. ATL is another transformation engine developed in the GMT project of Eclipse. The result of this study allowed us to choose ATL as the better option to implement the derivation guidelines. The derivation guidelines were expressed in declarative and imperative sentences, so the hybrid nature of ATL provides patterns and a language to specify heuristics of this kind. A transformation module was obtained as a principal result of this stage.

We consider the implementation of the transformation rules to be an important step in reducing the gap between the requirements models and the final software applica-

tion. In addition, as a part of our future work, we plan to integrate goal-oriented perspectives with the Communication Analysis method as an interesting alternative in order to offer more points of views to analyse organisational needs and to improve the MDD environment.

Acknowledgements

We thank to Giovanni Giachetti and Beatriz Marín for their support during the definition process of the metamodels, transformation rules, technological advices and scientific recommendations. In particular, we are grateful to Sebastian Bauersfeld for providing invaluable assistance and feedback about the research work presented in this paper.

References

- [1] I. The Standish Group International, "The Standish Group, CHAOS Summary 2010," ed, 2010
- [2] W. Bussen and M.D. Myers, "Executive information system failure: a New Zealand case study," *Journal of Information Technology*, vol. 12, issue 2, 1997, pp. 145-153.
- [3] S.T. Foster Jr. and C.R. Franz, "User involvement during information systems development: a comparison of analyst and user perceptions of system acceptance " *Journal of Engineering and Technology Management*, vol. 16, issue 3-4, 1999, pp. 329-348.
- [4] N. Juristo, A.M. Moreno, and A. Silva, "Is the European industry moving toward solving requirements engineering problems?," *Software IEEE*, vol. 19, issue 6, 2002, pp. 70-77.
- [5] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap." In: Conference on The Future of Software Engineering (ICSE'00), New York, USA, 2000.
- [6] G. Loniewski, E. Insfran, and S. Abrahão, "A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development." In: 13th International Conference, MODELS 2010, Oslo, Norway, 2010, pp. 213-227.
- [7] M. Ruiz, "A model-driven framework to integrate Communication Analysis and OO-Method," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, València, 2011.
- [8] S. España, A. González, and Ó. Pastor, "Communication Analysis: a requirements engineering method for information systems." In: 21st International Conference on Advanced Information Systems (CAiSE'09), Amsterdam, The Netherlands, 2009, pp. 530-545.
- [9] M. Ruiz, S. España, A. Gonzalez, and O. Pastor, "Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos." In: VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010), Jornadas de Ingeniería de Software y Bases de Datos (JISBD) Valencia, España, 2010, pp. 70-77.
- [10] K. Kronlof, *Method integration: concepts and case studies*. Chichester, UK.: John Wiley and Sons Ltd., 1993.

- [11] K. Kumar and R.J. Welke, *Methodology EngineeringR: a proposal for situation-specific methodology construction*. NY, USA: John Wiley & Sons, 1992.
- [12] M. Saeki, "A meta-model for method integration," *Information and Software Technology*, vol. 39, 1997, pp. 925-932.
- [13] G. Giachetti, "Supporting Automatic Interoperability in Model Driven Development Processes," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, Spain, 2011.
- [14] J.L. de la Vara, "Business Process-Based Requirements Specification and Object-Oriented Conceptual Modelling of Information Systems," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, Spain, 2011.
- [15] OMG. *Unified Modeling Language v1.5* 2003. Accessed 12-2005. Available: <http://www.omg.org/cgi-bin/doc?formal/03-03-01>
- [16] O. Pastor, S. España, and A. Gonzalez, "An Ontological-Based Approach to Analyze Software Production Methods." In: United Information Systems Conference (UNISCON 2008), Klagenfurt, Austria, 2008, pp. 258-270.
- [17] A. Gonzalez, S. España, M. Ruiz, and O. Pastor, "Systematic derivation of class diagrams from communication-oriented business process models." In: 12th edition of the Business Process Modeling, Development, and Support (BPMDS) series, in conjunction with CAiSE'11, London, Uk, 2011.
- [18] S. España, "Methodological integration of Communication Analysis into a Model-Driven software development framework," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, 2011.
- [19] S. España, A. Gonzalez, O. Pastor, and M. Ruiz, "Rules for the manual derivation of the Conceptual Model." ProS Research Centre, Universitat Politècnica de València, Valencia, Spain 2011. Available: <http://arxiv.org/abs/1103.3686v2>
- [20] S. España, A. González, Ó. Pastor, and M. Ruiz, "Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo." 2011. Available: <http://arxiv.org/pdf/1101.0105>
- [21] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T.J. Grose, *Eclipse Modeling Framework: A Developer's Guide*: Addison-Wesley Professional, 2003.