A Models-to-Program Information Systems Engineering Method*

Rene Noel[1,2][0000−0002−3652−4645], Ignacio Panach[3][0000−0002−7043−6227], and Oscar Pastor[1][0000−0002−1320−8471]

[1] Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València, Valencia, Spain
rnoel, opastor @pros.upv.es
[2] Escuela de Ingeniería Informática, Universidad de Valparaíso, Valparaíso, Chile
rene.noel@uv.cl
[3] Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València, València, Spain
joigpana@uv.es

Abstract. The Model-Driven Development paradigm aims to represent all the information system features through models. Conceptual-Model Programming offers a similar approach, but with a focus on automatic code generation. Both approaches consider modeling and traceability of different abstraction levels, where each level can be tackled with different modeling methods. This heterogeneity introduces a challenge for the quality of the traceability and transformations among models, especially when aiming for automatic code generation. In this paper, we introduce a holistic conceptual-model programming method to generate code from different abstraction levels (from the problem space to the solution space), through three modeling languages whose consistency has been ontologically ensured by two transformation techniques. Particularly, we focus on transformations from the strategic layer using i*, to business process layer using Communication Analysis (CA), and to the system conceptual model layer with OO-Method, which can automatically generate fully functional systems. Even though there are previous works that have proposed partial transformations among these modeling methods, this paper is the first one that deals with the perspective of putting together all the models in a single development method. For each transformation, we discuss what parts can be automatically performed and what parts need human intervention.

Keywords: modeling methods combination · model-driven interoperability · conceptual model programming.

* This project has the support of the Spanish Ministry of Science and Innovation through the DATAME project (ref: TIN2016-80811-P) and PROMETEO/2018/176 and co-financed with ERDF and the National Agency for Research and Development (ANID)/ Scholarship Program/ Doctorado Becas Chile/ 2020-72210494.

1 Introduction

The use of modeling languages for different information systems abstraction levels and the transformation between them, are key characteristics of model-driven approaches [13]. Most of the claims of these approaches, regarding improvements on product quality, process efficiency, and developer's satisfaction [18] are based on the suitability of the modeling methods and the quality of the transformations. However, combining modeling methods from different abstraction levels with different languages, semantics, and theoretical foundations is an open challenge: it is necessary to precisely define their connection to ensure the internal quality of the transformations, the quality of the models generated by the transformations, and the overall method quality, besides the quality of the independent methods [12].

This article presents a Models-to-Program Information Systems Engineering Method (M2PM), which combines three modeling methods for different abstraction levels, going from organizational modeling of strategic dependencies with i* [23], to business process modeling with Communication Analysis [6], and to an executable conceptual model of the system with OO-Method [19]. As its main contribution, this paper reinforces the feasibility of Conceptual-Model Programming paradigm [5] in practice, by showing that it is possible to design a holistic software production method that connects -with a sound methodological background- stakeholder's goals and requirements with their associated code. This is achieved by connecting in a precise way scientifically (but individually) validated methods and transformation techniques, ensuring traceability throughout the process and providing as much automation as possible.

The rest of the article continues with Section 2, where the related work is presented. An overview of the holistic modeling method and a detailed working example showing the model-to-program process is presented in Section 3. Finally, Section 4 details the conclusions and future work for the method.

2 Related Work

2.1 Connection of Modeling Methods

The model-driven community has widely studied the connection of models of different abstraction levels for developing information systems in the last decade. In a systematic literature review about interoperability [10], the authors identified several approaches for model-driven interoperability, i.e., the exchange of information among models. Model weaving regards the identification of semantic equivalences between the metamodels of the models to integrate, to generate specific maps between the concepts of the models. The pivotal metamodel approach is the equivalencies between metamodels identification using a reference metamodel to compare them; Pivotal ontology follows a similar approach. Meta-extensions are the transformations semantics additions to models to improve interoperability. Despite the early recognition of the interoperability approaches, the traceability among models is still an open challenge. In another literature

review [17], authors studied the state of traceability among models, identifying challenges related to the semantics of traceability and its generality. Most of the reviewed studies described problem-specific semantics for the transformations, which might be domain, organization, or project dependent.

With regard to the quality definition for the combination of modeling languages in a single method, Giraldo et als.[13] report issues both in the criteria for choosing the languages to be combined and in the overall quality assessment of the combined methods. The literature review concludes that there is a predominant subjectivity in selecting modeling languages, and raises questions about how it is assessed the suitability, coverage, pertinence, and utility of the languages to be combined. It also reports that most of the existing quality evaluation frameworks have definitions of a high level of abstraction, lacking implementation details, and are specific for a unique language combination. Authors continue this work in [11], presenting a method and a tool for a general quality evaluation framework, which helps to better define the quality concepts and the metrics for language comparison. One of the quality metrics supported by the framework regarding information loss is the preservation of constructs through the model transformations. Another quality metric regarding the suitability of a modeling language is the number of integration points that it provides for its connection with the other languages.

In summary, modeling language connection is still a challenge, although recent proposals approach systematically its quality evaluation. These additions offer insights about the desired characteristic for a new combination of methods, such as problem-independent traceability, transformations with a clear and defined interoperability approach, and focus on key quality attributes such as constructs preservation and the suitability of each modeling language.

2.2 Background

I* is an agent-oriented and goal-oriented modeling framework for the description and reflection of the intentionality of the actors of an organization. I* considers two modeling levels: the Strategic Dependency Model and the Strategic Rationale Model. In the Strategic Dependency Model (SDM), the actors are presented as nodes, and their intentions are represented by directed relationships from the actor that wants to achieve a goal to the actor that enables the achievement of the goal. These relationships are called dependencies, and there are four types of them: goal, soft-goal, task, and resource dependency. The Strategic Rationale Model (SRM) aims to detail the way that those dependencies are satisfied, linking the SDM dependencies to specific goals, soft-goals, tasks, and resources inside the boundaries of each actor. Those elements can be linked to represent task decomposition, means to an end, and contribution to soft-goals. The framework is implemented in OpenOME Requirements Engineering Tool [15]. The version 2.0 of the modeling language [3] is supported by the piStar Tool [20].

Communication Analysis (CA) [6] is an information system's requirements specification method, which allows business process modeling from a communication perspective. The model support three specification artifacts: the Com-

municative Event Diagram, the Communicative Event Specification, and the Message Structures specification. The Communicative Event Diagram graphically depicts the sequence of interactions between a Primary Actor (who starts the communication), a Support Actor (who is the organization's interface in the communication), and the Receiver actor (who is notified of the results of the event). The Communicative Event Specification allows the textual specification of requirements, through a template that considers contact requirements, the content of the communication, and the reactions produced after the communicative event. The Message Structure Specification allows to represent the information that is inputted, derived, or generated in the communication by defining one or more data fields, aggregations of data fields, and substructures. Aggregations are structures valuable to business logic, so they are also called Business Objects. The supporting tool for CA is a functional prototype based on Eclipse modeling Framework, the GREAT Process Modeler, described in [21].

The OO-Method (OOM) [19] is an automatic software production method from platform-independent conceptual schemes. It considers four views to model the information system: the structural view (Object Model), the behavioral view (Dynamic model), the logic (Functional Model), and the user interfaces (Presentation Model). OOM is based on the OASIS language [16] and considers a Conceptual Schema Compiler for the generation of platform-specific models and code. The supporting tool for OOM is INTEGRANOVA [2], that can generate fully functional web or desktop systems in many programming languages.

GoBIS [22] technique proposes nine guidelines for the derivation of CA models from i* models. The guidelines propose to derive a Communicative Event in CA for each dependency between i* actors, as well as to derive precedence of the events, although there is no temporal dimension in i*. Also, it promotes the generation of communication events for the registry of information of relevant actors. GoBIS guidelines are implemented in the tool GREAT Process Modeler, described in [21].

España proposes in [7, 14] the integration of CA models with OOM models, presenting a set of rules to derive OOM's object, functional, and dynamic models. The proposal considers rules that are fully automatable, as well as semi-automatic transformations that require manual modeling tasks from the analyst. This transformation technique is also implemented in the GREAT Process Modeler [21].

3 The M2P Information Systems Engineering Method

3.1 Method Overview

We introduce the Models-to-Program Information Systems Engineering Method (M2PM), as a connection of existing modeling methods and model transformation techniques. The goal of the method is to support the following claims:

- Improve the maintainability of the software product through traceability between the organizational level, business processes level, and the conceptual model of the system level.

– Improve the efficiency of the development process by providing as much automation as possible between the abstraction levels.

This is achieved by connecting in a precise way scientifically (but individually) validated methods and model transformation techniques. In Fig. 1 we present the connection of methods proposed: organizational modeling with i* [23], business process modeling with Communication Analysis (CA) [6], and Systems modeling with OO-Method (OOM) [19].

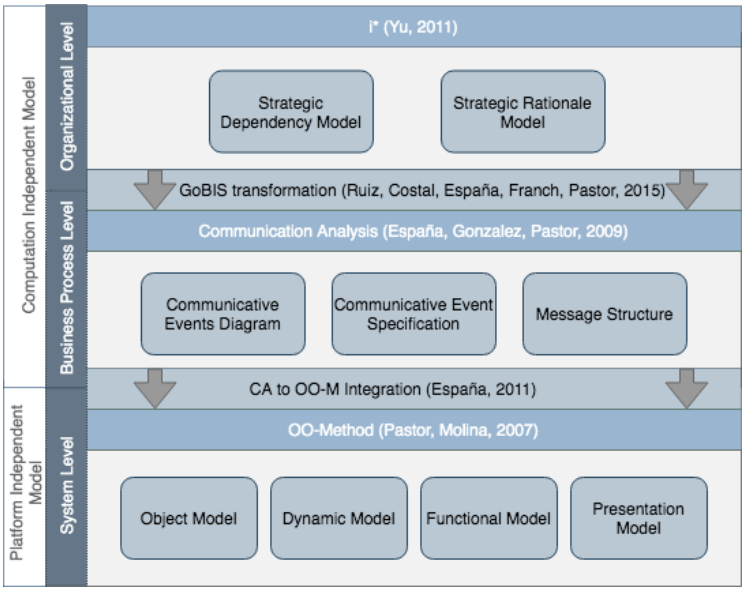

Fig. 1. Proposed connection of modeling methods.

As commented in Section 2, one of the key quality elements for method integration is its semantic consistency. We choose i*, CA, and OOM because of the ontological alignment of the existing transformation techniques [22, 7]. The transformation techniques use FRISCO [8] as a pivotal ontology to ensure the consistency of i* concepts with CA concepts, and of the CA concepts with the OOM concepts.

In the following subsections, we will present the models and transformations of the proposed method following an example. For each stage modeling level, we will provide the semantic justification for the transformations, as well as the rationale for mapping concepts from different abstraction levels, which are the basis for the successful combination of modeling languages [11, 17]. This ontological alignment led us to choose CA over other similar business process modeling methods, such as BPMN Collaboration Diagrams [1] and S-BPMN [9].

3.2 Working Example

For presenting the key elements of the method, we introduce the Custom Bicycle Company. The customers can order a custom bicycle, that is composed of a basic structure (that includes the frame, rims, grips, and chain) and one or many additional components, such as custom handlebars, tires, and pedals. Customers can choose a model for the basic structure (for example, a sports bicycle), its color, and size. For each of the components, customers can choose the color. The company must request the additional components to an external company. Once all the components are provided, the company delivers the bicycle to the customer.

3.3 Organizational Level Modeling with i*

The Strategic Dependency Model depicted in Fig. 2 represents the actors and their strategic dependencies. Circles represent the actors of the domain example: the customers, the clerk (who represents the organization before the external actors), and the provider of the components. The goals of the actors are pictured as an ellipse, and the direction of the relationship indicates that the source actors depend on the target actor to satisfy the goal. Resources dependencies are pictured as rectangles and represent that the source actor needs the resource from the target actor. Although i* provides a rich language to specify other types of dependency as well as how the goals are achieved, the example does not cover these aspects for the sake of simplicity.

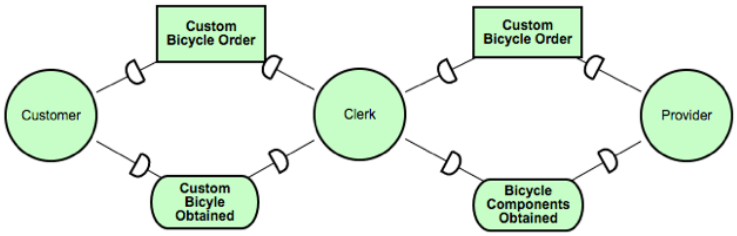

Fig. 2. Strategic Dependency Model.

3.4 Business Process Level Modeling with Communication Analysis

Transformation from i* model to Communication Analysis Model. The guidelines presented in [22] provide support for the transformation of i* models into Communication Analysis (CA) models. The transformation is not problem specific. The central idea of the transformation is that a strategic dependency of any type in i*, generates a communicative interaction between the same

actors in CA. In model transformation terms, organizational goals (represented in a source istar model) are materialized by business processes (represented in a target CA model). Resource dependencies that relate to information are transformed into a Communicative Event in CA, with all its associate concepts. The semantic of this transformation is as follows: for an Actor A to satisfy the need of an informational resource of an Actor B , the Actor A must communicate with Actor B to deliver the informational resource. Other types of dependencies also transform into communicative events with all their associated concepts, as previously detailed. It is noticeable that the semantic for each type of dependency is different. For instance, for goal dependencies the semantic can be expressed as: if an actor A depends on Actor B for achieving a goal, Actor B must communicate to Actor A the information that is relevant for Actor A to verify that the goal has been achieved.

The guidelines also support the transformation of actors from i* whose information is relevant for the business model, into a communicative event for the registry of its information. Other transformation supported by the guidelines deals with actors which satisfy the same dependency for two or more actors, which transform into a single communicative with a primary actor and many receiver actors. Also, subsequent dependencies between three or more actors are transformed into precedence relationships between the communicative events. The semantic of this transformation is: before Actor A can satisfy the need for the resource of Actor B, Actor C must satisfy the need of Actor A. This provides a sense of temporal precedence which is not explicitly modelled in i*.

It is important to note that some Communication Analysis Concepts cannot be generated following the guidelines. The structure of the input and output messages (that will be explained in the following subsections) must be elicited and documented by the Analyst. Also, the Support Actor for the Communicative Event must be chosen among the actors of the event, and alternate behaviors (known as Communicative Event Variants), cannot be derived.

Fig. 3 exemplifies the elements traced from the previous example to CA models, following the guidelines. In the example, the resource "custom bicycle order" (A in the left diagram of Fig. 3) and its source and target actors, clerk and customer, respectively, are mapped into a communicative event (A in the right diagram). This event has the following elements: customer as the primary actor (A.1) who starts the communication by sending an input message (A.2) to the clerk, who is the receiver actor (A.3) of the output message (A.4). The details of the communication are meant to be specified in the communicative event "customer places a bicycle order" (A.5).

The same transformation described above also applies to the dependency B in the diagram in the left of Fig. 3, which is transformed in the elements of the communicative event marked as B in the right. It is important to note that the Analyst can choose a more appropriate name for the message: although the content of the communication between the clerk and the provider can be the same "bicycle order", the aim of the communication of the clerk with the provider is to request the additional components, so the input and output messages are

named as "components order" (B.1 and B.2 in Fig. 3). This is the same case of the dependencies D and E in the diagram at the left in Fig. 3, which transforms into the communicative events D and E in the right of the diagram in the same figure.

The subsequent dependency of the resource "custom bicycle order" (C at the left) is transformed into the precedence relationship between the events "customer places a bicycle order" and "clerk places a component order" (C at the right of Fig. 3). Finally, regarding the registry of data of relevant actors, customer data would be valuable to represent in this way, but it is not depicted in Fig. 3 for simplicity.

As has been shown, almost all of the i* constructs are preserved in the CA model. Table 1 summarizes the elements that can be traced from i* to Communication Analysis, and what must be manually addressed by the Analyst.

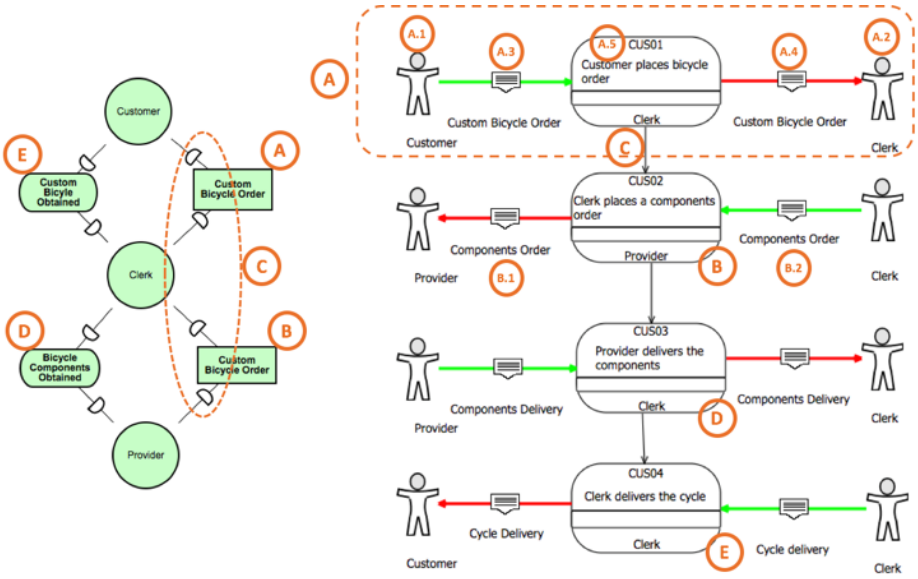

Fig. 3. Transformation from i* Strategic Dependency Model to a Communicative Event Diagram.

Additional Business Modeling Activities. Communication Analysis (CA) supports the specification of the communicative events, allowing the Analyst to specify requirements regarding the goal of the communicative event, the description, and a more detailed specification of contact, content, and reaction requirements. These elements are exemplified in Fig. 4for the communicative event "customer places a bicycle order".

Table 1. Traceability of concepts between i* and Communication Analysis.

CA Concept	I* Concept	Comment
Communicative Event	Goal, soft-goal, task, resource dependums	Traceable and semiautomatic.
Actors	Actors	Primary and receiver actors are derivable. Support actors cannot be derived.
Messages	Goal, soft-goal, task, resource dependums	Traceable. No details for message structure can be derived.
Precedence	Subsequent dependencies of the same dependum	
Communicative Event Specification	Not supported	-
Communicative Event Variant	Not supported	-

In the communicative event specification, the content requirements detail the information of the communication, in the form of a Message Structure (MS). The MS specifies the data elements in the communication, as well as other more complex structures, such as aggregations (structures containing one or more data fields or other structures), and iterations (several repetitions of the same field or structure). In the example, the MS "Bicycle Order" (colored in orange in Fig. 2 4) is composed of an initial aggregation (BICYCLEORDER, in red), which is composed of six data fields (number, date, and price of the order; model, size or color of the basic bicycle structure), and an iteration of several components (in violet). "Component" is also an aggregation, with two data fields (type and color). A special case of data field is customer (colored in brown): this is a reference field for other aggregations already defined in other communicative events. In the same way, the aggregations defined in this structure can be referenced in other communicative events. These aggregations are also known as Business Objects, given its value for the business process.

Finally, it is important to identify the supporting actor of a communicative event. Although GoBIS rules do not provide support for its automatic generation, the Analyst has the information to identify which of the two actors in the event belongs to the organization and set it as the supporting actor.

3.5 Conceptual Modeling of the System with OO-Method

Transformation from Communication Analysis Model to OO-Method Model. The rules in [7] allow the transformation of communicative events and message structures of CA into elements of the object model, functional model, and dynamic model of OO-Method (OOM). The transformation is not problem-specific. The presentation model of OOM is out of the scope of the transformation. In these rules, the Message Structures (as presented in Table 1) provide information for OOM's object model. The aggregations in the MSs (hereinafter

CUS01 Customer places a bicycle order

Goals: Customer orders a custom bicycle.
Description: Customer selects a basic structure and one or more additional components. Customer can select the color and size of the basic structure, and the color of each additional component.

Contact Requirements

Communication Channel: In person, by phone, by e-mail.
Temporal Restrictions: Only working days (09:00-18:00)
Frequency: 50 orders per week.
Required supports: Customer information.

Communication Content Requirements

Message Structure: Bicycle Order

Field	OP	Domain	Example
<BICYCLEORDER =			
{number +	g	number	43211
date +	i	date	01-05-2020
price +	i	money	250.5
customer +	i	customer	C0122,F.Miler
model +	i	text	Sport
size +	i	number	53
color +	i	text	silver
{COMPONENTS			
<COMPONENT=			
{type +	i	text	handlebar
color	i	text	black
}>}}>			

Structural constraints: Custom cycles must contain at least one cycle. **Contextual constraints:** Order Number is correlatively assigned and is unique.

Reaction Requirements

Treatments: The order is stored. **Linked Communications:** The Clerk is notified of the new order. **Linked behavior:** no exceptional behaviors are considered.

Fig. 4. Communicative Event Specification example.

namely Business Objects or BOs) transform into Classes in OOM's object model. The semantic of this transformation can be understood as: If the contents of the communication between two actors in the business process level are valuable, they must persist in the information system that supports the process.

Regarding the Communicative Events (CE), the primary receiver and support actor are transformed into Agents of the object model of OO-Method. Agents are classes that have execution permissions for the services of the classes. The services of the classes are derived from CEs, to allow the actors to create, edit, delete, and make complex operations with the objects, according to the behavior described in the communicative events. The semantics of these transformations can be interpreted as: If an actor has behavior associated with the content of the communication at the business process level, the actor must be able to execute the services that encapsulate that behavior at the system level. If the same BO is referenced in several communicative events, the transformation rules guide the generation of edit service; also, the rules support the generation of the logic to update the class, by introducing a valuation rule in OOM's functional model. For BOs that are referenced in several events, the transformation rules support the generation of OOM's dynamic model, where the states and transitions of each class are defined.

A special case are the BOs that change of state through the business process, which is also supported by the rules by generating the attributes, services,

and functionality to implement the state machine for the class. Finally, for communicative events in which the messages introduce changes for several BOs at the same time, the transformation rules support the generation of a transaction service in the object model, and a transaction formula in the functional model to specify all the services that must be connected in the transaction.

There are OOM's model elements that cannot be automatically generated following the rules and require manual modeling by the Analyst. For instance, the cardinality of some structural relationships in OOM's object model, the selection of data fields as unique identifiers of objects, as well as the null allowance and variability of attributes.

Fig. 5 presents examples for the most powerful transformations, that are commented below. Dealing with the generation of classes from Message Structures, the business object BICYCLEORDER (Fig. 5.A in the left) transforms into the class BicycleOrder (A in the diagram at the right). Data fields of the BO transform into attributes of the class. The referenced BOs, such as customer (left B), are transformed into structural relationships (right B). In this case, the customer is referenced but not defined, but, as was presented in the previous section, the guidelines provided support to generate a BO for the customer's relevant data. Regarding the nested aggregations defined in the MS, such as Component (left C), they are also transformed into structural relationships, and in this case, into a new class, with a cardinality given by the iteration (right C). Regarding the generation of services, when an actor introduces a new BO in the business process, i.e., the customer places a new bicycle order (left D in Fig. 5), it generates a creation service for the corresponding class (BicycleOrder), and the clerk, which is the supporting actor of the organization, gets access to this service as an agent (right D). If the bicycle order is referenced in several communicative events through the process (left E in Fig. 5), then an editing service is generated for the BicycleOrder class, and the valuation rules for registering the change of states of the object, in OOM's functional model (bottom right in Fig. 5).

As an example of a transaction service, if for the final delivery of the bicycle it is needed to register the "delivery date· in the BicycleOrder, and set the "last delivery date" attribute in the Customer (left F in Fig. 5), this would produce a transaction service in the BicicyleOrder class (right F).

Table 2 summarizes the elements that can be traced from i* to Communication Analysis, and what must be manually addressed by the Analyst. Most of the constructs of CA are preserved to OOM's models, including constructs originated in i* concepts. For example, the class BicycleOrder is traceable to a resource dependency in i*, offering strategic and business process context for the Analyst. This is a key contribution of the method combination proposal: the modeling methods and transformation techniques combination preserve most of the high abstraction level constructs, which gives strategic and business context to the system modeling, providing traceability and automation in the generation of the most important concepts in each level.

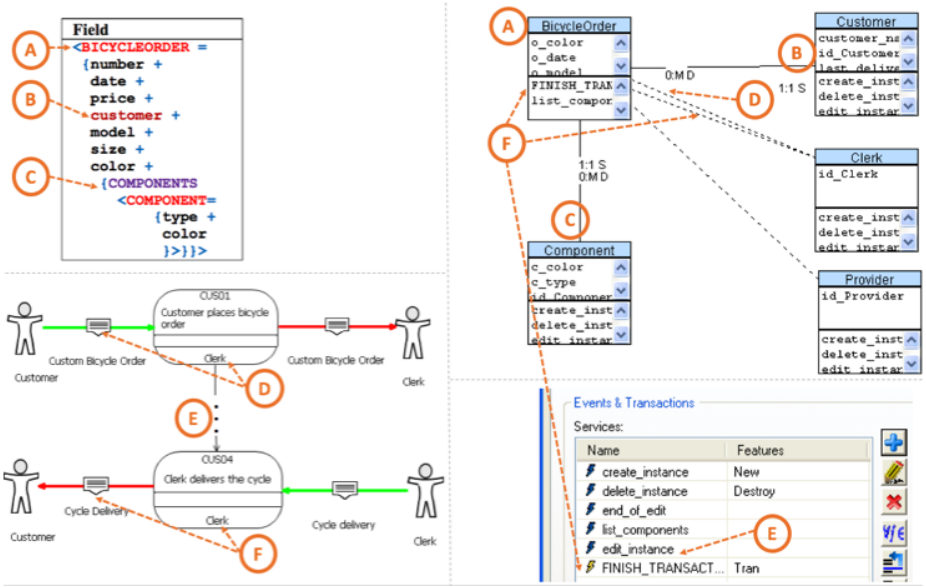

Fig. 5. Communicative Event Specification example.

Table 2. Traceability of concepts between Communication Analysis and OO-Method.

OOM Concept	CA Concept	Comment
Classes, attributes and relationships	Message structures	Traceable and automatic.
Services definition	Communicative events	Traceable and semi-automatic. Require some manual modeling and interpretation of text requirements.
Classes structural relationships	Message Structures, Communicative Events	Traceable and semiautomatic. Require some manual modeling and interpretation of text requirements.
Agents	Actors	Traceable and automatic.
States and transitions	Communicative Events and precedence.	Traceable and automatic.
Services logic (functional model)	Communicative events	Semi traceable and semiautomatic. Require some manual modeling and interpretation of text requirements.
Presentation model	Not supported	-

Additional system modeling activities. The transformation of CA models generates OOM's model elements with strong semantic foundations, however, there are OOM concepts that are not supported by the rules. Additional modeling is needed to fully specify the logic of additional services. OOM's presentation model cannot be derived from the rules in [7]. Due to the high technology readiness level of OO-Method and its tool support [2], fully functional software can be generated by using the embedded presentation patterns of the method. Also, complementary methodological approaches from models to working user interfaces can be introduced, such as the presented in [4].

4 Conclusions and Future Work

Information systems modeling at different levels of abstraction, and the transformations between them, are key elements of model-driven approaches to increase both product and process quality with respect to traditional methods. We presented the M2P Information Systems Engineering Method as the connection of three different modeling methods which are suitable for different abstraction levels: i* for organizational level, Communication Analysis for the business process level, and OO-Method for the system model level, which supports the generation of working code of the information system. The connection among these methods is supported by well-defined and empirically validated transformation techniques, that have been developed separately, and have never been put together in a holistic method. With an example, we showed the feasibility of conceptual model programming in practice, emphasizing the semantics of the transformations, and the support provided by the method for the Analyst to complete the models of each abstraction level.

Regarding the limitations of the example, certainly many special cases are not covered (for instance, exception handling in business process models), and the scalability of methods that work in small examples is a matter of concern. However, the example goal is to demonstrate the feasibility of the method and identify traceability and automation issues that are also present in more complex problems. Further work aims for analyzing improvement opportunities to connect concepts that could be not currently considered by the transformation techniques, to enhance traceability and provide as much automation as possible. Also, modeling challenges for improving the overall quality of the modeling method, such as overlapping of modeling activities, must be identified.

References

1. Business Process Model and Notation (BPMN), Version 2.0 p. 532
2. Integranova Software Solutions, http://www.integranova.com/es/
3. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. arXiv preprint arXiv:1605.07767 (2016)
4. Díaz, J.S., López, O.P., Fons, J.J.: From user requirements to user interfaces: A methodological approach. In: International Conference on Advanced Information Systems Engineering. pp. 60–75. Springer (2001)

5. Embley, D.W., Liddle, S.W., Pastor, o.: Conceptual-model programming: a manifesto. In: Handbook of Conceptual Modeling, pp. 3–16. Springer (2011)
6. España, S., González, A., Pastor, Ó.: Communication analysis: a requirements engineering method for information systems. In: International Conference on Advanced Information Systems Engineering. pp. 530–545. Springer (2009)
7. España Cubillo, S.: Methodological integration of Communication Analysis into a model-driven software development framework. Ph.D. thesis (2012)
8. Falkenberg, E.D.: A framework of information system concepts: the FRISCo report. University of Leiden, Department of Computer Science, Leiden (1998)
9. Fernández, H.F., Palacios-González, E., García-Díaz, V., G-Bustelo, B.C.P., Martínez, O.S., Lovelle, J.M.C.: Sbpmn—an easier business process modeling notation for business users. Computer Standards & Interfaces **32**(1-2), 18–28 (2010)
10. Giachetti, G., Valverde, F., Marín, B.: Interoperability for model-driven development: Current state and future challenges. In: 2012 Sixth International Conference on Research Challenges in Information Science (RCIS). pp. 1–10. IEEE (2012)
11. Giraldo, F.D., España, S., Giraldo, W.J., Pastor, O.: Evaluating the quality of a set of modelling languages used in combination: A method and a tool. Information systems **77**, 48–70 (2018)
12. Giraldo, F.D., España, S., Pastor, O.: Analysing the concept of quality in model-driven engineering literature: A systematic review. In: 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS). pp. 1–12. IEEE (2014)
13. Giraldo, F.D., España, S., Pastor, O., Giraldo, W.J.: Considerations about quality in model-driven engineering. Software Quality Journal **26**(2), 685–750 (2018)
14. González, A., España, S., Ruiz, M., Pastor, Ó.: Systematic derivation of class diagrams from communication-oriented business process models. In: Enterprise, Business-Process and Information Systems Modeling, pp. 246–260. Springer (2011)
15. Horkoff, J., Yu, Y., Eric, S.: Openome: An open-source goal and agent-oriented model drawing and analysis tool. iStar **766**, 154–156 (2011)
16. Lopez, O.P., Hayes, F., Bear, S.: Oasis: An object-oriented specification language. In: International Conference on Advanced Information Systems Engineering. pp. 348–363. Springer (1992)
17. Mustafa, N., Labiche, Y.: The need for traceability in heterogeneous systems: a systematic literature review. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). vol. 1, pp. 305–310. IEEE (2017)
18. Panach, J.I., España, S., Dieste, O., Pastor, O., Juristo, N.: In search of evidence for model-driven development claims: An experiment on quality, effort, productivity and satisfaction. Information and software technology **62**, 164–186 (2015)
19. Pastor, O., Molina, J.C.: Model-driven architecture in practice: a software production environment based on conceptual modeling. Springer Science & Business Media (2007)
20. Pimentel, J., Castro, J.: pistar tool–a pluggable online tool for goal modeling. In: 2018 IEEE 26th International Requirements Engineering Conference (RE). pp. 498–499. IEEE (2018)
21. Ruiz, M.: TraceME: A Traceability-Based Method for Conceptual Model Evolution. Springer (2018)
22. Ruiz, M., Costal, D., España, S., Franch, X., Pastor, O.: Gobis: An integrated framework to analyse the goal and business process perspectives in information systems. information systems **53**, 330–345 (2015)
23. Yu, E.: Modeling strategic relationships for process reengineering. Social Modeling for Requirements Engineering **11**(2011), 66–87 (2011)