Supporting organisational evolution by means of model-driven reengineering frameworks

Marcela Ruiz, Sergio España, Óscar Pastor
PROS Research Centre
Universitat Politècnica de València
Valencia, Spain
{lruiz, sergio.espana, opastor}@pros.upv.es

Arturo González
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Valencia, Spain
agdelrio@dsic.upv.es

delivery life-cycle activities. For instance, the Object Management Group (OMG) is working on promote an industry consensus on modernisation of existing application by means of the initiative named Architecture-Driven Modernisation (ADM) [3]. This initiative is based on the MDD paradigm and it has been implemented in an Eclipse-based tool named MoDisco [4]. There are several modernisations projects; ADM establishes a general scenario for modernisation. Nevertheless, full support by means of methods and tools for the software reengineering process is still an open challenge [5]. The model management community have proposed different approaches to support evolution and they highlights the importance of models in this activity[6]. Moreover, no complete frameworks to support enterprise information system evolution have been proposed yet.

This paper, presents a research in progress that is aiming at getting the most out of MDD proposals to support information system evolution in organisational domains. We conceive business process models as an important artefact for representing organisational behaviour. Since, there are already several works that cover the derivation of information systems from/to business processes (reverse engineering or forward engineering) [7-9]. We focus on developing specific artefacts for evolving business process models.

A previous work has presented a general view of our proposal, a MDD organisational reengineering framework [10]. In short, we have been inspired by the metaphor of a “horseshoe” of Kazman et al [11]. Carrying the horseshoe metaphor to the MDD field, reengineering can provide an interesting maintenance framework for organisational domain. Briefly, the reengineering framework consists of three processes and four artefacts. The first process is the reverse engineering process; whose input is the first artefact, the As-Is system (current system). The result of the reverse engineering process is the second artefact, the As-Is models (that represent the As-Is system in an abstract way). The second process is the evolution process; whose input are the As-Is models. As a result, the output of the evolution process is the third artefact, the To-Be models (evolved models). The third process is the forward engineering process; whose input is the To-Be models and the output is the fourth artefact, the To-Be

---

1 The ERCIM News 88 special theme was “Evolving Software”[2]. The magazine put together a set of papers to give an overview of both traditional and emerging software engineering techniques, tools and approaches used by software evolution experts.

---

I. INTRODUCTION

Continual organisational change and associated transform in processes and products must be considered as a fundamental rule of competitive strategy for continued progress [1]. Evolution of enterprise information systems can be confronted following the model-driven development paradigm (MDD). This way, infrastructures, technologies, staff, business processes, etc. can be viewed from perspectives in high level of abstractions. Moreover, with regard to the keynote of the ERCIM News 88 magazine1, some of the external drivers for changing information systems are innovation, cost reduction and regulation; external factors that need be supported by techniques, tools and methods. Several proposals and tools are currently aiming at supporting post-
system (system that results from the reengineering process and fulfils the new goals and needs of the organisation).

With regard to the evolution process, this paper faces the open challenge of providing support for model-driven and pattern-driven model evolution. Fig. 1 presents a big picture in order to locate our contributions.

Fig. 1. MDD reengineering framework. Focus on evolution process

The main contributions of this paper are the following:

- We provide a pattern definition metamodel in order to specify reengineering patterns and we explain it use. The objective of this artefact is to guide model evolution in organisational context.

- We provide an evolution metamodel to establish traces in order to indicate changes between both As-Is models and To-Be models. The objective of this artefact is to store model evolution.

The paper also presents a case that illustrates the proposal and demonstrates its feasibility. Moreover, the benefits and limitations of our proposal are discussed. As a proof of concept, we use communicative event diagrams, a communication-oriented business process modelling language [12]. Other business process modelling languages (i.e. UML activity diagrams [13] or business process modelling notation (BPMN) [14]) in any case, can be used as well.

The paper is structured as follows: Section II reviews related research. Section III describes business activities behaviour and presents the pattern metamodel. Section IV presents the evolution metamodels. Section V presents an illustrative example. Section 0 discusses lessons learned, our conclusions and future work.

II. RELATED RESEARCH

This section reviews related research in order to exemplify alternatives to support the model evolution process in a model-driven reengineering framework.

A use-case model, an object model, and traceability links between these models were defined in [15]. This work proposes the evolution of these models as “changes in use-case model”, where a use-case model consists of use-case diagrams, activity diagrams and object models (class diagrams and sequence diagrams). Traceability links are defined as relations between the elements of use-case models and object models. In short, the idea is to maintain the traceability among diagrams at different levels of abstraction and at different stages of evolution (As-Is system and To-Be system).

Research in model weaving attempts to handle fine-grained relationships among elements of different models. Relationships conform to a metamodel that specifies link semantics [16]. Due to the complexity of establishing links among models, the weaving metamodel could be adapted to support different domains (i.e., requirements models and goal models).

From the point of view of business process modelling, business process models change according to the requirements of an application domain. The work in [17] provides an infrastructure that supports the adaptation of both process modelling languages and process models (specifically, a multi-level metamodelling framework). The idea is to provide a metamodel to specify a standard for process modelling language and to specify the changes based on the new requirements in the language. The standard process modelling language is aligned with its specialisations (domain-specific process modelling language). In this way, all language definitions will be based on the same metamodel and will share a common set of modelling constructs.

Model management confront problems in many databases application domains (e.g. data warehousing, semantic query processing, meta-data management, meta-data integration, schema evolution etc.); research projects in this area are aiming at providing high-level abstractions artefacts in order to offer a generic solution [18-19]. As a result, there are works in model management to deal with manipulation of models [20]. Hence, representations of models and model mappings are proposed as formal specifications in high-level algebraic operations. Bernstein [21] presents a full description of all of the model management operators. For instance, the operator match takes two models as input and returns a mapping between them. In [21], mapping between models is defined as mapping between objects or combination of objects. The matching process highlights the equality or similarity of concepts between the models that have been matching. In addition, in [22] is presented a proposal for model management, where, formal descriptions or models are manipulated by means of mappings or relations among them. The idea is to define the semantics of the operators...
in enough detail to face three model management activities: model integration, model evolution, and round-trip engineering.

In summary, the related research mentioned above supports parts of the model evolution process. Although, supporting model evolution by means of model-driven techniques is an open challenge, these related research provide scientific advances to be reused. Specifically, we study the model management operators proposed by Bernstein [21]; and as a proof of concept of this paper, we explore possibilities for evolving business process models in the sense that its deals with organisational reengineering domains.

III. SUPPORTING MODEL EVOLUTION: DESIGN OF PATTERN DEFINITION METAMODEL

España [8] has carried out a revision about communicative behaviour. This revision focus over communicative events and it highlights communicative events that can be revised by both analyst and the users. These communicative events can lead changes in business processes specification.

Due we are aiming at supporting model evolution; we were inspired by España’s revision. Hence, we identify a set of business activities behaviour that analyst can revise in order to improve the current business process. As an initial attempt, we identify the following business activities behaviour:

• Exception in internal treatments: A business activity that involves an authorization can demand an exceptional behaviour related to approvals or decision taking.
• Deviations from optimist assumptions: A business activity related to carry out orders of resources can lead saturation of resources and it can affect mediator objects that are responsible of a specific management (i.e. saturation of resources, saturation places, over booking, supplies, etc.). As a result, deviation from optimist assumptions can affect basic business activities and it can demand exceptional behaviour related to approvals in order to maintain resource balance.
• Exceptions in external treatments: Exceptions in business activities outside the range of action of organisational system or environment. This case is related to a business activity that depends of feedback from a business activity outside range of action of organisational system. These exceptions in external treatments can imply a decision taking in order to offer different alternative.
• Issuance audit: Business activities related to issue specific documents can demand an audited output. Consequently, business activities related to issuance can lead exceptional behaviour in order to ensure issuance.
• Reception audit. As before business activity, reception audit is related to confirm reception of information. Thus, business activities related to reception of information can lead exceptional behaviour in order to ensure reception.

- Audit of occurrence: A business activity needs the occurrence of other business activity. It can demand an exceptional behaviour related to ensure the occurrence of a specific business activity.
- Audit of information content: A business activity affected by indicators of the business rules can demand an exceptional behaviour related to decision taking.
- Audit of normative events: Business activities related to normative sequence of events defined in the business rules. It can demand an exceptional behaviour where business activities should be reordered in order to fulfil the business rules.

We think that business activities behaviour mentioned above can be part of a body of knowledge to guide business processes evolution. In order to specify business activities behaviour, we propose to take advantage of the concept of pattern. “A pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution million times over, without ever doing it the same way twice” [23]. Even though pattern definition was thought in the context of buildings and constructions, software engineering community has used this concept [24]. In order to bring a pattern-driven specification for business processes model evolution, we were inspired in the patterns metamodel for system engineering presented by [25]. As a result, we propose the metamodel presented in Fig. 2.

![Fig. 2. Pattern definition metamodel](image)

The PATTERN metaclass can have multiples keywords (KEYWORD metaclass) to allow its localisation. In addition, PATTERN can have related bibliography or documentation (BIBLIOGRAPHY metaclass). Also, it is recommendable to specify the author information in order to create contacts or networking to build the pattern repository (AUTHOR metaclass). We advise to specify one example (at least) in order to clarify it use (EXAMPLE metaclass). EXAMPLE metaclass is pointing to MODEL metaclass via two references in order to represent both relation As-Is model and To-Be model. This way, we suggest to exemplify each pattern by means of a description, As-Is and To.Be models. MODEL metaclass is painted gray because is an external metaclass. For instance, BPMN and Activity Diagrams
of UML are models that could be used as example for the specification of a pattern. Also, PATTERN metaclass can have a set of alias (also-known-as) (ALIAS metaclass).

In order to illustrate, we use the pattern metamodel presented in Fig. 2 to specify two business activity behaviours: a) exception in internal treatments and b) issuance audit. We choose it because we will use it in section V (the illustrative case).

TABLE I. SPECIFICATION OF THE PATTERN OF EXCEPTION IN INTERNAL TREATMENTS

<table>
<thead>
<tr>
<th>PATTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Creation date:</td>
</tr>
<tr>
<td>Problem description:</td>
</tr>
<tr>
<td>Goal description:</td>
</tr>
<tr>
<td>Discovery guidelines:</td>
</tr>
<tr>
<td>AUTHOR</td>
</tr>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Email:</td>
</tr>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Email:</td>
</tr>
</tbody>
</table>

TABLE II. SPECIFICATION OF THE PATTERN OF ISSUANCE AUDIT

<table>
<thead>
<tr>
<th>PATTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Creation date:</td>
</tr>
<tr>
<td>Problem description:</td>
</tr>
<tr>
<td>Goal description:</td>
</tr>
<tr>
<td>Discovery guidelines:</td>
</tr>
<tr>
<td>AUTHOR</td>
</tr>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Email:</td>
</tr>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Email:</td>
</tr>
</tbody>
</table>

Due to the pattern metamodel is not defined for a specific modelling language; we use the communicative event diagrams, a modelling technique of the Communication Analysis method to exemplify the pattern for business activity behaviour of exception in internal treatments [12]. On the other hand, to exemplify the pattern for business activity behaviour of issuance audit, we chose UML activity diagrams (see TABLE II.).

Afterward, we propose to store pattern specification in a pattern model repository (see Fig. 1). This way, when an analyst analyses a business process, he/she may require the application of a pattern. Then, analyst chooses patterns from the pattern model repository to apply it to the business process according to its behaviour. Analyst can create new patterns to support business process behaviour that have not been previously considered.
IV. SUPPORTING MODEL EVOLUTION: DESIGN OF EVOLUTION METAMODELS

This section presents our proposal to manage model evolution. Basically, we propose a metamodel that support both traceability between As-Is models and To-Be models, and recording pattern application. To build the evolution metamodel, we were inspired in solutions proposed for model management and model traceability in model-driven engineering context.

We were inspired by Bernstein [21], besides, even though we do not pretend to match two models, we consider as an interesting idea mapping representation as an intermediate model. Similarly, we adopt mapping model definition and we establish mapping between objects as a relationship between objects from different models by means of mapping operators (similar, equal, etc.). To illustrate, Fig. 3 presents a mapping example between model A and model B. MAPPING 1 says that CONCEPT A1 and CONCEPT B1 are been mapping by means of mapping operator equal (=), while MAPPING 2 says CONCEPT A2 and CONCEPT B2 are been mapping by means of mapping operator similar (≈).

Fig. 3. Mapping between model A and model B (This figure is an adaptation taken from [21])

On the other hand, traceability is a concept widely used in MDD. In this paper, we will use the following definition for traceability: traceability is a relationship between entities belong to different models (adaptation taken from [26-27]). There are proposals in order to support traceability among models. Besides, traceability could be considered as a model. Jouault [28] presents a simple trace metamodel able to specify relationships among elements without have into account a specific context (i.e. model to model transformation, model to text transformation, etc.).

In order to provide an evolution metamodel, we borrow from Bernstein research some mappings operators and its semantics. In this article, we call mapping operators as evolution operator because its role in an evolution process to indicate changes in As-Is models and To-Be models. In particular, we analyse the operator match; accordingly, we propose an evolution metamodel that takes advantage of traceability metamodel proposed by Jouault in order to provide artefacts to store traceability information. We extend this traceability metamodel following the specification for carrying out model mapping presented by
Bernstein. Fig. 4 shows our evolution metamodel. It contains an EVOLUTION_TRACE metaclass owing an operator attribute to store information about evolution. We conceive an enumerator (EVOLUTION_OPERATOR) to specify four basic evolution operators:

- The first basic evolution operator is **equal**. By equal, we mean that two model elements are alike.
- The second basic evolution operator is **modified**. By modified, we mean that two model elements are alike but some properties could be changed. For instance, the property name of a business process could be changed.
- The third basic evolution operator is **added**. By added, we mean that a model element has been added.
- The fourth basic evolution operator is **deleted**. By deleted, we mean that a model element has been deleted.

Fig. 4. Evolution metamodel

EVOLUTION_TRACE metaclass is pointing to MODEL_ELEMENT metaclass via two references in order to represent both As-Is and To-Be relationships from/to EVOLUTION_TRACE metaclass. MODEL_ELEMENT metaclass is an external metaclass (this metaclass is gray in order to indicate that it is part of the metamodel that has been linked with traceability metamodel). In order to apply the proposed pattern definition metamodel (see Fig. 2), we define a PATTERN_APPLICATION metaclass. Basically, this metaclass consist of an abstraction to relate patterns definition with a set of evolution traces. PATTERN_APPLICATION metaclass is pointing to PATTERN metaclass and it is pointing to EVOLUTION_TRACE metaclass. PATTERN metaclass is an external metaclasses from pattern definition metamodel. The objective of PATTERN_APPLICATION metaclass is recording evolution traces when a pattern is applied. This way, we can provide a support to store the set of evolution traces when a pattern is applied; for instance, the information about evolution traces can be recorded in a text log. Finally, the evolution operator are established by means of an enumerator (EVOLUTION_OPERATOR <<enumerator>>)

To illustrate, Fig. 6 presents an instance of the evolution metamodel. We use communicative event diagram metamodel as business process model [12] (in order to facilitate well understanding, we provide a view of this metamodel in Fig. 5).

Fig. 6 describes the model evolution process of expense report business process model; which corresponds with the example presented in TABLE I. Although our evolution process support evolution of a set of models, in order to simplify, in this example we describe the evolution traces to evolve from one As-Is model to one To-Be model. Note that evolution traces whose attribute is equal do not have association with pattern application. We advise do not store information related to equal evolution operations because do not provide useful information about changes in the To-Be models. However, we provide support for equal evolution operator in the case of the analyst considers it use. We advise to provide enough information in order to provide complete log or reports about models changes; concretely the attribute description of the PATTERN_APPLICATION metaclass.

Fig. 5. View of communicative event diagram metamodel
V. ILLUSTRATIVE CASE

This section presents an application of our proposal for evolving business process models (see Fig. 1). The idea is to carry out a laboratory demo (a.k.a. lab-demo) to illustrate the use of the proposed metamodels (patterns definition metamodel, evolution metamodel; Fig. 2 and Fig. 4 respectively) and the use of the patterns repository. As a practical case, we use Communicative Event Diagrams (CED), a business process modelling technique that adopts a communicational perspective and facilitates the development of an IS that will support those business processes [8, 12]. This modelling language is involved into Communication Analysis method, a communication-oriented business process and requirements engineering method to specify information systems. We present in this paper an adaptation of the SuperStationery Co. case. SuperStationery Co. is a company that provides stationery and office material to its clients. The company acts as an intermediary: the company has a catalogue of products that are bought from suppliers and sold to clients. This case is presented in full detail in [29]. In this paper, we focus on part of the sales manager business process (acronym SALE, see Fig. 7).

Fig. 6. Instance of evolution metamodel
receiver actor of the communicative event SALE 1. An ingoing relationship is a communicative interaction that feeds the IS memory with new meaningful information. The main direction of the outgoing communicative interaction is from the primary actor to its related communicative event. For instance, the relationship named order between the primary actor client and the communicative event SALE 1 is an ingoing communicative interaction. An outgoing relationship is a communicative interaction that consults the IS memory. The main direction of the outgoing communicative interaction is from the communicative event to its related receiver actor. For instance, the relationship named assigned order that is between the communicative event SALE 2 and the receiver actor supplier is an outgoing communicative interaction. The precedence relationships are represented as arrows among communicative events (e.g. SALE 2 requires the previous occurrence of SALE 1).

B. The evolution process of sales manager business process of SuperStationery Co. case

The As-Is model (see Fig. 7) relates that to place an order, most clients phone the sales department, where they are attended by a salesman (this process is related to the communicative event SALE 1). Then the client requests products. The salesman takes note of the order. Then the sales manager assigns the order to one of the many suppliers that work with the company (this process is related to the communicative event SALE 2). The supplier authorises the order (this process is related to the communicative event SALE 3).

To carry out the evolution process, we can follow two basic steps: The first step consists of analyse the pattern repository to trace the behaviour of the business process with the problem description of the patterns. The second step is the application of the pattern by means of the evolution metamodel. Hence, analysing the As-Is model, we can suggest an evolution process focus on the behaviour of the communicative event SALE 3 because it involves an internal treatment that can be traced with business activities behaviour analysed in the TABLE 1. This way, following TABLE 1, we analyse both problem description and goal description to relate communicative event SALE 3 with this pattern. We analyse the discovery guidelines suggested by the pattern to formulate questions below: Is the order always authorised? It is absolutely impossible that an order is rejected? As a result, we analyse the example provided by the pattern specification and we decide to apply the pattern. In this case, we apply the evolution metamodel to use evolution traces. Thus, we start to record each use of evolution traces. For instance, when an order is rejected, the sales manager assigns the supplier to the order to evaluate it again (loopback). In this case, we add an or element in the To-Be model to indicate that SALE 2 can occur after the occurrence of SALE 1 or it can occur after the occurrence of the event variant SALE 3.1 (see the diamond that precedes communicative event SALE 2). Evolution traces with background gray are related to pattern application and they are stored to create an evolution log. In order to simplify the

Fig. 7. Communicative event diagram of part of the sales manager business process of SuperStationery Co. case (As-Is model)

A. Concepts of Communication Analysis: Communicative Event Diagrams

To facilitate understanding of the illustrative example, this subsection presents a brief explanation of the concepts used for Communication Analysis. We are focus on communicative event diagrams.

A communicative event is the organisational action that is triggered as a result of a given change in the world that is intended to account for that change by gathering information about it. It is a set of actions that are related to information (acquisition, storage, processing, retrieval and/or distribution), which are carried out in a complete and uninterrupted way. The unity criteria allows communicative events to be identified [30]. Each communicative event is represented as a rounded rectangle and is given an identifier, a number and a descriptive name (e.g. SALE 1 in Fig. 7). For each event, the actors involved are identified. The primary actor triggers the communicative event and provides the input information. For instance, the client is the primary actor of the communicative event SALE 1. The interface actor is in charge of physically interacting with the IS interface. Interface actors are specified at the bottom of the event. For instance, the salesman is the interface actor of the communicative event SALE 1. The receiver actors are those who need to be informed of the occurrence of an event. The sales manager is the

---

For a more comprehensive understanding, additional context and details might be necessary. The diagram illustrates the communicative event flow for the SuperStationery Co. case, detailing the roles of actors such as the client, sales manager, and supplier, and their interactions, including the processes of placing an order, assigning an order, and authorising the order.
example, we do not draw each association among evolution traces and pattern application. In contrast, we provide a rectangle that groups the set of evolution patterns associated with pattern application. However the equal evolution operator does not provide meaningful information about the evolution process, for illustrative purposes, we specified it to register the business activities that persist after evolution process.

To conclude, due to the application of the pattern repository and the evolution metamodel is not dependent with the modelling language, we can use this approach to support model evolution.

VI. LESSONS LEARNED AND FUTURE WORK

Reengineering field has been faced by MDD paradigm. Enterprises can confront information system evolution taking advantage of different level of abstractions that model perspective can bring. This current research effort aims at providing a reengineering framework that gets the most out of the existing proposals in MDD, in order to support organisational improvement and information system maintenance. Our proposal intends to provide a support for carrying out reengineering processes in organisational domains. This way, we face information system maintenance from the benefits that the maturity of MDD can provide. Furthermore, MDD can provide reengineering frameworks of benefit as methods, tools, infrastructures, etc. These factors can improve technological transfer to enterprises. Nevertheless, methods, tools, infrastructures etc. for organisational evolution is still an open challenge in the MDD field.

This paper, presents our proposal to support the evolution process in an organisational reengineering domain. As a result, we propose a set of artefacts to support model evolution. As a first artefact, we conceive a set of business behaviour to identify common performance of business activities. Consequently, these business activities behaviour is specified by means of a pattern definition metamodel. In addition, we provide a pattern model repository to store patterns specifications. This pattern metamodel is specified without a specific business process modelling language. As a second artefact we conceive a metamodel evolution. The objective of this metamodel is to guide pattern application. Traceability capabilities are provided in order to trace possible changes between both As-Is model and To-Be model. As a result, we provide artefacts that are pattern-oriented and model oriented to support model evolution.

Fig. 8. Evolution traces for sales manager business process of SuperStationery Co. case
We have already apply our proposal in a laboratory demo in order to presents the use the artefacts to evolve business process. As a practical case, we use communicative event diagrams, a communication-oriented business process modelling technique involved in the Communication Analysis method. Our proposal is specified in a general manner; this way, reengineering framework is not inclined to a specific modelling language.

Currently, we are working in the implementation of pattern definition metamodel and evolution metamodel; also we are analysing a technological support for pattern model repository. We are analysing possible notations to specify model evolution. In addition, we are searching for templates to specify traceability and model evolution that results after an evolution process; concretely, we want to record the evolution process information in logs.

For future works, we are planning a validation of quality and understanding of metamodels as a first step; later we want to validate the sensitivity of the proposal. In addition, we will plane to complement metamodel evolution with metamodels that guide this evolution based on intentional/goal metamodels. This way, we are aiming at facilitating to trace enterprise company desires with its information system evolution.

ACKNOWLEDGMENT

We acknowledge the support of the Spanish Ministry of Science and Innovation project PROS-Req (TIN2010-19130-C02-02), the Generalitat Valenciana project ORCA (PROMETEO/2009/015), the Santiago Grisolía grant (GRISOLIA/2011/005) and the ERDF structural funds.

REFERENCES