Abstract—Companies need to rethink business processes, infrastructures, technologies, staff, etc. according to new demands, strategic goals, and changes in their environment. Information System evolution has been supported by the reengineering process. Since the models are now part of an increasing number of engineering processes and model-driven software development is a well-established paradigm to support information systems, the reengineering process could be improved with the incorporation of the model-driven development paradigm. This paper presents a model-driven organisational reengineering framework to support organisational improvement. The main idea is to get the most out of the existing proposals in model-driven development for the reengineering process (reverse engineering, improvement processes, and forward processes). Methods, techniques, model-driven tools, and perspectives for analysing information systems at a high level of abstraction are presented as a part of this proposal. A reference framework for the reengineering process is established to provide a consistent set of concepts that support the proposed framework. An illustrative example is introduced to present the feasibility of the proposal. The example illustrates the SuperStationery Co. case in order to present a reengineering process to align an existing desk application with a web application that fulfils the strategic goals of the company. Finally, we conclude with an analysis and discussion about lessons learned and future works.

Keywords— reengineering process; model-driven software development; improvement process; model evolution; requirements engineering; model transformation; goal-oriented requirements engineering.

I. INTRODUCTION

Since we are living in a changing world, companies need to rethink business processes, infrastructures, technologies, staff, etc. according to the new demands of their environment or changes in their organisational objectives. Therefore, organisations must often adapt to an evolution that can be focused on structural organisations, business processes, information technologies (IT), and/or organisational goals. Business processes should also be transformed to support the new processes and tasks that result from the involvement of new objectives or goals in the organisation. Therefore, the organisation structure should be analysed, and the staff of the company must be taken into account as playing an important role in the organisational evolution.

Similarly, information technologies are part of the evolution of the organisation. Both, software systems and hardware systems should be part of the innovation processes of the organisations. However, even though hardware systems are important and are also impacted by technological changes, this work only deals with the evolution of software systems. For software systems, the high pressure of a very short time-to-market often forces developers to implement the code of the application directly, without using a disciplined development process, which may have disastrous effects on the quality and documentation of the delivered software application [1]. These practices have been the motivation for opening new research lines in order to support post-delivery life-cycle activities. Reengineering is a concept that has been widely used in software maintenance. Reengineering corresponds to the process of reverse engineering to get higher level specifications, an evolution of these specifications, and forward engineering to develop a new version of the software application [2-3].

Within the area of software development, the Model-Driven Development (MDD) paradigm has emerged as a successful trend. This paradigm defines characteristics that establish software development that is focused on models rather than computer programs as a primary product [4]. Since models are expressed using concepts that are less bound to the underlying implementation technology and are closer to the problem domain, there are open research lines in MDD. Specifically, there are several works in MDD reengineering for the maintenance of legacy software. The support of legacy software has been the principal reason for innovating in reengineering processes and for offering different solutions to adapt legacy system [5-6]. For instance, Zhao et al present a MDD approach for reengineering legacy software systems to evolve them to web service applications [7]. Briefly, the MDD approach is integrated into a reengineering process where models play a role beyond the conventional design and documentation.

By reintroducing organisational evolution, reengineering could provide an interesting maintenance framework. In this work we present a framework that takes into account current software that obtains the models of software application through a reverse engineering process. These models are evolved according to new requirements and changes that arise from the organisational environment. These new models are the input for a forward engineering process, where a classical
technique (Waterfall, Incremental & Iterative, etc.) could be used to obtain a software system that is adapted to the new environment.

Business process models, goal models and organisation structure models are the main artefacts that we use to represent the models of the current system and the models of the desired system. Since an alignment of these models is necessary, an improvement process should support the model evolution to specify the requirements that will be reflected in new models that will define the desired system. In this way, system requirements are analysed from a requirements and goal perspective.

As our main contributions, we provide the following:

- A consistent set of concepts and terminology about the reengineering process and its artefacts.
- A set of proposals to exemplify alternatives to support parts of the reengineering process following the model-driven paradigm.
- A model-driven organisational reengineering framework provides artefacts and tools to carry out the processes that are part of it.

We develop an illustrative example to present the feasibility of the proposal. A final web application is presented as a result of the reengineering process.

The paper is structured as follows: Section 2 presents a reference framework for the reengineering process. Section 3 reviews related works. Section 4 presents the model-driven organisational reengineering framework (specifically, an analysis of the improvement process). Section 5 presents an illustrative example. Section 6 discusses the results, some lessons learned, our conclusions, and future lines of work.

II. A REFERENCE FRAMEWORK FOR THE REENGINEERING PROCESS

This section establishes a consistent set of concepts and terminology about the reengineering process and its artefacts. The aim is to facilitate the analysis and comparison of related works and set the theoretical foundations for a model-driven organisational reengineering framework.

There are many different definitions of the reengineering process, depending on the field of application or the nature of the evolving system. For instance, in the field of business process reengineering, Hammer and Champy have defined the reengineering process as "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance such as cost, quality, service, and speed"[8]. The definition proposed by Arnold [3] is widely used by the software engineering community, "reengineering is the whole process of reverse-and-forward engineering".

Several works define the reengineering process as having three basic processes (reverse engineering, improvement, and forward engineering) and four basic artefacts (As-Is system, As-Is models, To-Be models, and To-Be system). Based on these definitions, we can establish the following general definition for the reengineering process. The reengineering process is the whole process, which consists of a set of activities aimed at evolving systems from a current situation (As-Is system) to a future, improved/desired situation (To-Be system): 1) the As-Is system is abstracted as a model (As-Is model); 2) an analysis on the current needs is performed in order to improve the system’s current limitations, and the changes of the As-Is models are reflected in a new improved set of models (To-Be models), 3) the To-Be system is developed based on the To-Be models.

Figure 1 presents an outline of the reengineering process. The three basic processes (reverse engineering, improvement, and forward engineering) are represented by means of arrows that link the four basic artefacts (As-Is system, As-Is models, To-Be models, and To-Be system) that are represented by means of rectangles. Note that the As-Is system and the To-Be system can exist at different moments in time, so they can be considered to represent either two distinct systems or a single system that has evolved. Following Figure 1, we establish the definition of the three processes and the four artefacts that conform the reengineering process.

The first basic process is the reverse engineering process. Reverse engineering is the process of analysing the current system to identify the system’s components and their interrelationships and create representations of the system in another form or at a higher level of abstraction (we adapted this concept from [10]). When the current limitations or risks of a system make a reengineering process advisable, then the current system is analysed in order to identify disconnects (e.g. inconsistencies between the functions of a process that cause the existing process to fail to achieve its goals) [9]. The first artefact is the As-Is system. The As-Is system is the current system that needs to be aligned with the new requirements of the environment and is the input of the reverse engineering process. The system’s abstraction is part of the reverse engineering process and is represented by means of models. The creation of the models is guided by the use of the various modelling methods available (business process models, activity diagrams, conceptual models, etc.). The system’s models allow time, costs, resources etc. to be analysed. This analysis highlights the concepts and relationships that need to be reengineered [11]. The result of the reverse engineering process
is the second artefact, the **As-Is models**. The As-Is models are a set of models that represent the As-Is system at a high level of abstraction. The As-Is models are the artefacts to evolve.

The second basic process is the **improvement process**. The improvement process aims at identifying alternatives to the current situation (represented by means of As-Is models) that satisfy the strategic goals or needs of the desired system. Thus, the input of the improvement process is the As-Is models. The improvement process has several names; restructuring [10] and producing of alternatives [11] are two of them. The improvement process is an iterative process that leads to the creation of the models that represent the desired system by means of modelling methods [12]. In short, the As-Is models are analysed and aligned with the goals and needs of the desired system. As a result, the output of the improvement process is the third artefact, the **To-Be models**. The To-Be models are a set of models that result from the improvement process and, thus, represent the desired system at a high level of abstraction.

The third basic process is the **forward engineering process**. The forward engineering process is the process of moving from high-level abstractions and logical implementation-independent designs to the physical implementation of a system (we adapted this concept from [11]). The input of the forward engineering process is the To-Be models, and the output is the implemented system, which hopefully corresponds to the desired system that is delimited by the goals and needs of the environment. Therefore, the fourth artefact is the **To-Be system**, which is the system that results from the reengineering process. The To-Be system fulfils the goals and needs of the desired system.

In practice, there are some recurring general themes: most reengineered processes tend to align with several jobs that are combined into one; decision-making falls to the workers and not the managers; process steps are performed logically and naturally; checks and controls are reduced or eliminated; hands-off are minimized; there are single points of contact[8].

### III. RELATED WORKS

This section presents a set of proposals to exemplify alternatives to support parts of the reengineering process following the model-driven paradigm. Nowadays, there are different proposals to support parts of a reengineering process with techniques and methods.

MoDisco [12] is a model-driven reverse engineering proposal that offers an open source solution. MoDisco uses Model-Driven Engineering (MDE) principles and techniques. Specifically, the reverse process is composed of two phases: Model Discovery and Model Understanding. Model Discovery is a process that obtains a model that represents a view on the legacy system from its source code, raw data, available documentation, etc. The Model Discovery phase could be conducted to develop several models. Model Understanding is the process where the models obtained from the Model Discovery process are analysed. The idea is to exploit these models in order to achieve the final desired representation of the system. MoDisco has been implemented as an Eclipse open-source project to develop model-driven tools. Depending on the reverse engineering objectives, MoDisco offers support to manage source code, data metrics, visualisation, documentation, etc. The use of metamodels from the Object Management Group (OMG) standard is provided (Knowledge Discovery Metamodel (KDM) and Software Metrics Metamodel (SMM)).

A full model-driven development framework for the forward engineering process is proposed by España [13]. The proposal analyses the information system from a communicational perspective. The requirements models resulting from the analysis of the information system are the input for deriving the OO-Method conceptual models[14]. The forward process is supported by derivation techniques that could be implemented manually by a human analyst or by means of the Atlas Transformation Language (ATL) model transformation that automates this task[15]. The OO-Method conceptual models are supported by Integranova [16], which is a model compiler that automates the derivation of full software applications (user interface, data bases, behaviour, etc.) from OO-Method conceptual models. In this way, the software development process is covered from requirements to generation of functional prototypes.

Different proposals provide methods and techniques to support model evolution in the improvement process. In several proposals for managing model evolution, the traceability among models is important to be able to control application versioning and to link elements of different levels of abstraction (requirements, concepts, source code, etc.). Managing and evolving models are important activities that are supported by traceability links among models.

There are many proposals that support evolution in object-oriented development. A use-case model, an object model, and traceability links between these models were defined in [17]. 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 [18]. 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 [19] 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 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.

In summary, there are several research projects that support parts of the reengineering process. Proposals in reverse engineering and forward engineering processes could be adopted by frameworks to support organisational reengineering processes. There are gaps for supporting model improvement processes using the model-driven paradigm. Tools, method and guidelines are necessary to offer a stable framework to support organisational improvement and the demands of organisational evolution (indicators, simulation, requirements implementation, goal analysis, etc.).

At this point, we focus on the importance of the As-Is models and the To-Be models. We relate the previous works in order to analyse the gaps. Our analysis shows that there are model-driven proposals to support some parts of the reengineering process, but a complete solution with tools and method is missing. MoDisco is a reverse engineering proposal, that guides the generation of models at a high level of abstraction from source code. The Model Understanding process of MoDisco is a phase where the models are improved in order to fulfil the requirements of the To-Be system. However, requirements models and goal models that represent the As-Is system are missing to analyse the current situation. Also Modisco does not offer alternatives to specify the To-Be system.

Proposals in model evolution and model weaving show how to support traceability among models. Traceability among models at different levels of abstraction and traceability among models at different states of evolution (As-Is models and To-Be models) are necessary artefacts to be able to manage and control the evolution and improvement of a set of models. There are proposals that support traceability in conceptual models, but business processes and the evolution of enterprises are represented in requirements that need to be classified from different perspectives. Therefore, conceptual models represent the static behaviour of the system. The process perspective, communicative perspective, and goal perspective are views of the system that aim at specifying information about the system.

IV. A MODEL-DRIVEN ORGANISATIONAL REENGINEERING FRAMEWORK

In this section, we introduce a model-driven organisational reengineering framework. Two objectives of an organisational reengineering are to produce one or more alternatives to the current situation that satisfy the strategic goals of the enterprise and to adopt innovative practices. Most organisations need to map the existing process first (the As-Is models) in order to design new processes. A more thorough analysis of the As-Is models highlights the gaps and the deficiencies that impede the fulfilment of the goals and needs that motivate a To-Be system. For this reason, we propose to analyse the As-Is system from communicative and goal perspective by means of two modelling methods: Communication Analysis and i*.

Communication Analysis is a communication-oriented business-process modelling and requirements method that proposes the analysis of information systems from a communicational perspective [20]. This method is currently being applied in complex projects in industrial environments. In addition, Communication Analysis is supported within a model-driven software development framework. This framework could be viewed as a forward engineering process because of the generation of source code from requirements models. Communication Analysis defines the Communicative Event Diagram (CED), which describes information-related actions that are identified by the unity criteria and are carried out in a complete way following an external stimulus. The Event Specification Template (EST) structures the requirements associated to a communicative event. Among other requirements, EST contains a description of new and meaningful information that is conveyed to the IS in the event, which is specified using Message Structures (MS). Many of the constructs defined by the method are defined through a metamodel that supports specification. Communication Analysis models are the input of a model transformation strategy to derive OO-Method conceptual models, which is an object-oriented, model-driven development framework that is implemented in Integranova [13, 16]. The OO-Method conceptual models could be compiled in the Integranova modeller to automatically generate source code. Communication Analysis covers software development and support models to represent business processes and requirements by means of MS and CED. However, support for managing goal models is missing.

The i* framework [21] is a well-consolidated goal-oriented approach that allows IS to be modelled in a graphical way, in terms of actors and dependencies with each other. The use of i* is appropriate for modelling intentional concepts and the evaluation of alternatives. There is a method to support the reengineering process that focuses on the goal perspective, where i* models are used to represent the systematic generation of alternatives[22].

The model-driven development community has a challenge in the business process reengineering field, which implies to provide a model-driven support for an organisational reengineering process. In answer to this challenge, we propose a model-driven organisational reengineering framework to support organisation improvement. The general view of the reengineering process that is presented in Figure 1 was adapted to specify the As-Is models and the To-Be models in terms of the requirements and goal perspectives (see Figure 2). The As-Is models are the result of an analysis of the current system. A set of changes in the As-Is models leads to the creation of the To-Be models. Traceability among perspectives is taken into account (see dashed lines). The improvement process is from As-Is models to To-Be models. The traceability among these sets of models is necessary to analyse the changes between the As-Is system and the To-Be system (see horizontal dashed lines). For organisation structure models, the hierarchical diagrams, documentation, and information related to the staff of the organisation should be analysed and related to the requirements models. The organisation staff is an important artefact to consider when analysing the requirements models and goal models. For Communication Analysis requirements models, CED and MS are the models that represent the
business processes and the static and dynamic information of the system. With regard to the goal perspective, the i* models represent the intentional aspects.

![Diagram](image)

**Figure 2. Reengineering framework to support organisational improvement**

The model-driven paradigm offers several advantages, such as the possibility to develop model-driven tools to manage and support As-Is models and To-Be models. Since the model-driven paradigm can get the most out of a set of models, the ideas presented in related works could be considered as strategies to support the improvement process. There are metamodel proposals to support traceability among metamodels, and these ideas could be applied in this framework. The reverse engineering proposal implemented in the MoDisco tool is supported by Eclipse [23]. Communication Analysis requirements models are also supported by Eclipse and are integrated into the model-driven software development tool IntegraNova.

Nowadays, we are analysing a project to align Communication Analysis requirements models with i* models is being analysed. The idea is to align both models with a reference ontology (i.e. FRISCO). Consequently, a complete alignment between the goal perspective and the communicative perspective can be obtained, and all models will be interconnected.

V. ILLUSTRATIVE EXAMPLE

This section present an application of the proposed framework presented in Figure 2. The idea is to apply a lab demo to illustrate the use of the proposed perspectives (communicative and goal perspectives) and the existing tools. The case presented in this paper is 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 [24]. In this paper, we focus on part of the sales manager business process (acronym SALE). Communication Analysis requirements models and i* models are used to specify the As-Is models and the To-Be models.

A. Concepts of Communication Analysis requirements models and the i* model

To facilitate understanding of the illustrative example, this subsection presents a brief explanation of the concepts used for Communication Analysis (Communicative Event Diagrams and Message Structures) and i*.

1) Concepts of Communicative Event Diagrams

The Communicative Event Diagram is a business process modelling technique that adopts a communicational perspective and facilitates the development of an IS that will support those business processes [13, 20].

A communicative event 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 [25]. 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 Figure 5). 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 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 ingoing 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 order that is between the communicative event SALE 1 and the receiver actor sales manager is an outgoing communicative interaction. The precedence relationships are represented as arrows among communicative events (e.g. SALE 1 requires the previous occurrence of PROD 2 and CLIE 1).

2) Concepts of Message Structures

Message Structures is a specification technique that allows the message that is associated to a communicative interaction to be described [26].

A substructure is an element that is part of a message structure. This way, LINE, Client and Payment type are substructures that are part of the substructure ORDER (See Figure 7 a). There are two classes of substructures: fields and complex substructures. A field is a basic informational element of the message that is not composed of other elements. A data field is a field that represents a piece of data with a basic domain. For instance, payment type is a data field. A reference field is a field whose domain is a type of business object. For instance, Client refers to a client that is already known by the IS.
3) Concepts of i* model

Actor boundaries indicate intentional boundaries of a particular actor. All of the elements within a boundary for an actor are explicitly desired by that actor [27]. For instance, the sales manager is an actor (see Figure 4). A goal represents an intentional desire of an actor. For instance, an increase in sales is an intentional desire of the sales manager. The specific way to satisfy the goal is described through task decomposition. For instance, advertising products in TV, providing online order support and providing loyalty programs are tasks that aim at satisfying the goal of increasing sales. A means end link indicates the relationship between an end and a means to attain it. For instance, to increase sales (the goal), the means to achieve it is providing strategy sets. A task decomposition represents the different decompositions of an element. For instance, the task named simplify logistic is decomposed in the task allow many destinations.

B. The reengineering process of the SuperStationery Co. case

To place an order, most clients phone the sales department, where they are attended to 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. The supplier can either accept or reject the order.

The company has identified increase sales with respect to the previous year as a new goal for this year. The processes are analysed in order to determine how to improve the business processes in terms of efficiency and to determine what kind of strategies could be applied. Therefore, the As-Is system must be analysed. Figure 3 presents the current desk application that supports placing an order (communicative event SALE 1). Some clients have several destinations where they require stationery products. The desk application presented in Figure 3 shows that the system does not have the support to manage several destinations. As a result, the clients must place one order for each destination.

Analysing the As-Is system and the goals that the enterprise intends to achieve, we can justify carrying out a reengineering process of the As-Is system in order to provide a To-Be system that is aligned with the goals of the enterprise. Hence, we use our framework (see Figure 2). The desk application presented in Figure 3 could abstract a CED to analyse the communications among the organisational actors and processes.

Since the main goal of the company is to increase this year’s sales with respect to the sales of the previous year. Different strategies could be carried out. Figure 4 presents the i* model which represents the goals and tasks that are intended by the company.
The analysis of SALE 1 shows that the interface actor (the actor that is in charge of editing messages on the interface of the information system) is the salesman. Since the intention is to provide online support to place an order, a CED of the To-Be system is designed so as to align the Sales Manager process with the i* model (see Figure 4). Consequently the CED is modified: the interface actor of the SALE 1 must be the client since this actor will place the order in the online application.

In order to fulfil the task simplify the logistics, the communicative event SALE 1 must support the definition of many destinations of products on a single form. This modification of the process could minimize reprocessing (placing one order for each destination). This happens when a client has different places to send product. Therefore, the MS of SALE 1 must support many destinations on the order form. Figure 7 presents the MS of the As-Is system without the support for many destinations. The Figure 7 b presents the support for many destinations. By means of a set of model transformation, it is possible to carry out a forward engineering process to obtain a prototype of the application that supports the To-Be system. The support for model transformation from Communication Analysis requirements models to OO-Method conceptual models has been presented in our previous work [28] [29]; the derivation of the class diagram is described in [28], and the derivation of the states machine diagram is described in [29].
presents information that is not specified in the MS that is presented in Figure 7 (e.g. the field Planned delivery date). This occurs because the Class Model that was compiled in Integranova specifies the complete case of SuperStationery Co.

We were motivated by the possibility of getting the most out of the existing model-driven proposals to improve the reengineering process. For this reason, in this work we have proposed a model-driven organisational framework to support organisational improvement. The framework itself follows the model-driven paradigm (e.g. metamodel conformance); we have proposed methods, techniques, and model-driven development tools. We have also proposed a consistent set of concepts about the reengineering process and its artefacts in order to define the most appropriated model-driven organisational reengineering framework. The concepts were defined in a reference framework for the reengineering process. This reference framework establishes each concept of the reengineering process (e.g. the As-Is system, the reverse engineering, the As-Is models, the improvement process, the To-Be models, the forward engineering, and the To-Be system). A reference framework facilitates the analysis and comparison of related works and guides the theoretical foundations for the model-driven organisational reengineering framework proposed in this paper.

We have focused on the importance of the As Is models and the To-Be models. Support creation, evolution and analysis of the As-Is models motivate projects in the model-driven development research area. We propose to analyse the information systems from different perspectives. Specifically, we have taken into account two perspectives, communication perspective and goals perspective; both supported by the Communication Analysis method and the i* framework. Communication Analysis is a communication-oriented business process modelling and requirements method that proposes to analyse the information system from a communicative perspective. This method provides business process and requirements models to analyse the information system from a static and behavioural point of view. i* framework is a well consolidated goal-oriented approach that allows IS to be modelled in a graphical way, in terms of actor and dependencies with each other. The systematic generation of alternatives and indicator analysis could be done using this framework.

We provided an illustrative example (SuperStationery Co), in order to facilitate understanding of the proposed framework. Modifications in business process models, goals models, and static information is necessary to achieve the goals of the company. We used our framework to analyse the As-Is system and to specify some strategies in order to achieve the company goals that motivate the To-Be system. We provided an instance of the framework models (As-Is and To-Be models) to show existing technological and metamodel support for the modelling activities. As a result, we exemplified the reverse engineering, the improvement process and the forward engineering, and we presented the results of each process. We presented an i* model and a communicative event diagram with its corresponding message structures. We used tools metamodel-compliant that highlight the main role of the model-driven paradigm. As a result of the lab-demo, we presented a web application resulting of the reengineering process. This web application is in compliance with the needs of the environment. We carried out a comparison between the As-Is system (desk application) and the To-Be system (web application).

As future work, we intend to provide traceability support among the different levels of abstraction, the perspectives, and the evolved models. In addition, we want to provide an ontological alignment between Communication Analysis requirements models and the i* framework. We are currently working on this alignment; the results will facilitate the traceability between the communicative perspective and the goal perspective. Since technological support for the model-driven reengineering framework is also necessary, we are currently analysing existing metamodel support for the models specified in the proposal. We are planning to do a survey in the Valencia community to analyse the current reengineering process and to determine what kind of methods, tools, and strategies should be implemented. The results will improve the proposed framework presented here. So that a model-driven organisational reengineering framework with methods, tools, and use guides can be used in the real world.
AKNOWLEDGEMENTS

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


