Business Performer-Centered Design of User Interfaces

Kênia Sousa and Jean Vanderdonckt

Abstract. Business Performer-Centered Design of User Interfaces is a new design methodology that adopts business process (BP) definition and a business performer perspective for managing the life cycle of user interfaces of enterprise systems. In this methodology, when the organization has a business process culture, the business processes of an organization are firstly defined according to a traditional methodology for this kind of artifact. These business processes are then transformed into a series of task models that represent the interactive parts of the business processes that will ultimately lead to interactive systems. When the organization has its enterprise systems, but not yet its business processes modeled, the user interfaces of the systems help derive tasks models, which are then used to derive the business processes. The double linking between a business process and a task model, and between a task model and a user interface model makes it possible to ensure traceability of the artifacts in multiple paths and enables a more active participation of business performers in analyzing the resulting user interfaces. In this paper, we outline how a human-perspective is used tied to a model-driven perspective.

1 Introduction

User Interfaces (UIs) have been historically specified, designed, developed, and tested according to multiple design methodologies that have reflected a particular emphasis that was prevalent during a certain period. Each design methodology always adopted a certain viewpoint to drive the development life cycle. Over years, we witnessed the following viewpoints:

Kênia Sousa
Université catholique de Louvain, Louvain School of Management, Belgian Laboratory of Computer-Human Interaction, Place des Doyens, 1, 1348 Louvain-la-Neuve, Belgium
e-mail: kenia.sousa@uclouvain.be

Jean Vanderdonckt
Université catholique de Louvain, Louvain School of Management, Belgian Laboratory of Computer-Human Interaction, Place des Doyens, 1, 1348 Louvain-la-Neuve, Belgium
e-mail: jean.vanderdonckt@uclouvain.be

H. Hussmann et al. (Eds.): MDD of Advanced User Interfaces, SCI 340, pp. 123–142.
springerlink.com

Data-centered design: in this methodology, an interactive system is considered as a structured collection of data that will result into a data model, from which one or many UIs could be produced in order to properly manage this collection of data. The main advantage of this methodology was that deriving a UI for a particular data structure was straightforward. But the main disadvantage was that each data structure was then associated to a single screen in which the user has to carry out multiple functions that were not clearly identified [1]. It was then the responsibility of the end-user to properly execute the functions as expected in his/her tasks, thus resulting in several mismatches. In addition, data models did not initially consider the relationships between the concepts. In this case, the different interaction objects are derived from the information contained within the model, for instance, considering only the attributes of entities. It was clear at that point of the research that they needed to consider modeling functions and tasks [1].

Function-centered design: in this methodology, an interactive system is considered a collection of functions working on data identified in the aforementioned data model, that will result into a domain model. A domain model departs from a data model in that not only it contains the relationships between the data structures (e.g., as associations in a UML class diagram), but also its related constraints, and the functions associated to these data (e.g., the methods in a UML class diagram). The main advantage of this methodology was that deriving a UI for a particular function was possible to generate not only the layout of an interface, but also its dynamic behavior. The drawback of this approach was that only primitive functions were mainly supported (e.g., create, read, update, delete, list, search, print), thus forcing the end-user to re-interpret a task to be carried out in terms of such functions. Discovering, locating, and executing these functions was considered as a burden for end-users [2]. Therefore, researchers also acknowledged the need to consider task models for user interfaces with highly complex and flexible dialog structures.

Task-centered design: in this methodology, an interactive system is considered to support a collection of inter-related tasks that represent the user perspective of work, not the system perspective of work, thus resulting into a task model for every task of interest. A task model characterizes the way end-users carry out their effective task, not the prescribed tasks as found in manuals or procedures. In this way, it is expected that such a task model conveys a way of working that is close to the end-user. The main advantage of this methodology is that a UI derived from a task model is assumed to fit the end user's purpose since it should reflect his/her way of working. The difficulty of this approach is that many different UIs could be derived from the same task model and there is not enough effective knowledge today available that guides this derivation in order to ensure the quality of the result [3]. In addition, even though with some initiatives to consider user roles for a specific set of tasks in multi-user applications [4], it does not necessarily differentiate how user's profiles, skills and other aspects impact and differentiate these tasks and the design of user interfaces.

User-centered design: in this methodology, an interactive system is designed by gathering requirements directly from the end-user and by modeling these

requirements in such a way that they can drive the rest of the development life cycle to maintain the end-user's perspective. Several ways exist in order to establish and follow a user-centered design, such as task modeling (as in the task-centered design), user modeling (e.g., through a user model), participatory design, contextual inquiry [5], etc. The main advantage of this methodology is that the UI resulting from the process is expected to be as close as possible to the end-user's expectations, abilities, and preferences. The difficulty with this methodology is that there is a wide range of instruments in order to conduct a user-centered design and, even with a high probability of success, there is no guarantee that a usable UI will be obtained. Indeed, a good instrument could be used in an inappropriate way. A second difficulty is that one task is examined at a time, thus posing some challenges in how to organize tasks in time and space. Indeed, an end-user is rarely working alone in his/her context of use. Rather, the end-user is incorporated in a series of interconnected tasks, some of these tasks could be collaborative, cooperative, competitive, or coopetitive. Therefore, there is a need to proceed with the user-centered design in order to encompass aspects that were not captured in the task model or in the traditional user-centered design in the context of complex business process applications.

Process-centered design: this methodology uses the business process model as "a structured, measured set of activities designed to produce a specific output for a particular customer or market" [6] in order to consider complex contexts. Different from the previous methodology, it does not start from the end-user's viewpoint. Although there are proven UI design methodologies, such as in user-centered design, process-centered design [7] differentiates itself by precisely focusing on business process intensive software which has not been the case with other UI design methodologies. More recently, a data-centered approach for business processes called artifact-centric business process models [8] puts business artifacts at the center of the approach. This approach focuses on key business-relevant entities, their lifecycles, and how and when services are invoked on these artifacts. It has concepts that are similar to traditional data models and business process models, but with augmented data records that correspond to business-relevant artifacts and richer semantic expressions for task conditions and consequences. Some recent works in this area propose aligning business processes with UIs with direct links between activities in processes and elements on the UIs. However, such strategies leave aside the consideration of the user interaction, which is much richer than what is specified in business processes. For this purpose, we introduce the following methodology:

Business Performer-centered design: in this methodology, it considers business processes, while not neglecting benefits brought by user-centered design by giving equivalent importance for business process performers (or simply business performers), who are, at the same time, end-users of enterprise systems [9]. This dual and balanced consideration is particularly challenging because end-users have been historically ignored by business process notations and methodologies by favoring a top-down approach when designing enterprise systems. Additionally to considering user-centered design on this proposal, we not only emphasize the global aspects of

the work, but also individual aspects of each end-user, which is more precise than the attempt to summarize the human aspects using roles. Our main effort was to find out how user-centered design would be formally represented in this approach and, at the same time, keeping the compatibility with existing business process models. In our case, we propose that this merging point is a task model that serves as the cornerstone: the task model should be derived from the business processes that will in turn produce one or many UIs corresponding to the derived tasks. In this way, the expected win is that when a business process will change, the task model will change accordingly and so does the UI model corresponding to this task model, implementing a consistent alignment between business and user interfaces, and, consequently, providing traceability.

Considering these different viewpoints, several design methodologies have been defined over time with their specific purposes and aiming to be more thorough than the previous ones (see Fig. 1). In a timeline, Trident [10] automates user interface design from data models. In Mecano [2], each class in the hierarchy of the domain model is assigned to a window. In a task-based approach, a task model is given to identify UIs, for instance, CTT [11] recommends a task model where the interactive task is recursively and hierarchically decomposed into sub-tasks; and the UsiXML methodology [3] based on the Cameleon Reference Framework [12] has task and domain models at its center to specify user interfaces independent of implementation. In user-centered design, cognitive models of human users, known as user models, are used to predict human error and learning time, and can thus serve as a cheaper alternative to user testing, such as CogTool-Explorer [13]. Diamodl [14] takes a process centered design, which advocates to adapt BP modeling to include aspects of task modeling using BPMN for both BP and task modeling. While MDHI [15] is a design methodology and tool for the user interface life cycle that has business modeling as a starting point, including the business data. Finally, the UI-Business Alignment (or Usi4Biz) [9] considers business processes and their performers (end-users) to manage enterprise systems.

The remainder of this chapter is organized as follows: Section 2 will define the business performer-centered design as a way to align business process and UIs and introduces the concepts that are underlying to this design methodology. Section 3 will detail how this design methodology has been applied in a corporate environment, i.e. a very large telecommunication company, and how model-driven engineering has been used in order to support applying this methodology. Section 4 will discuss the potential advantages of this model-driven engineering as expected or observed in this case study, as well as identified shortcomings. Section 5 will conclude this chapter by highlighting the main contribution and by presenting some avenues of this research.

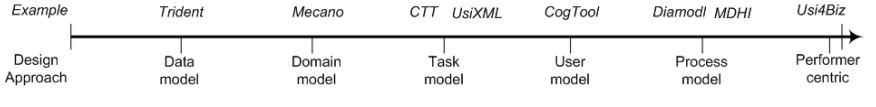

Fig. 1 Timeline of approaches

2 Going towards Performers and Users

One of the main goals that are aimed with the evolution of methodologies is to go beyond individual users or groups within a single organization. Emerging methodologies aim to consider the complex behaviors between people, among groups, among organizations, and between organizations and their environments. Fig. 2 depicts another perspective of the evolution of these different paradigms.

Despite the evolutions on these approaches and on the needs to address more complex behaviors, recent works [16, 17] still focus much more on how the artifacts are linked, handled and maintained than on the people who actually conceptualize and those who use enterprise systems. Thus, they end up failing to consider that these people are actually the ones who are key to making the organization work. Therefore, with a more human perspective, the business performer-centered approach considers the complexity of these relations intrinsic to business processes and adds the viewpoints of the business performers as active agents.

With that purpose, the UI-Business Alignment Methodology [9] considers business performers as active agents who open a new channel for business improvement by informing any sort of issues they face with systems. These identified issues may lead to changes on the processes, on the UIs or on both.

We rely on epistemological research to turn the recurring focus from business analysts and place it on business performers. This is in accordance with Giddens duality of structure theory [18], in which the organizational structure contains the behavior of professionals, but their behavior also makes the structure possible because these professionals have and apply their knowledge when acting, which may then change the organizational structure. Applying this theory to our methodology, business performers must adhere to the corporate structure following business processes, but they also bring changes to such structure since they think and behave differently.

Concerning the first state of the dual structure, in order to be able to adhere to defined processes, business performers must understand well how to perform these processes. However, it is common to find business processes defined at a high level that does not represent the tasks they should perform in their daily work. That is an important consideration to have task models with a lower level of granularity to specify performers' tasks. This way, there are fewer deviations from the tasks assigned to business performers. That is a reason for the growing interest from

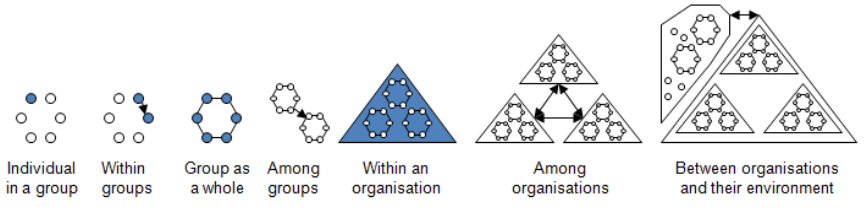

Fig. 2 Evolution of approaches

organizations to know how specific agents, not generic roles, accomplish their work [19]. Therefore, we focus on actual users that we can identify within the organization since they are the ones who actually perform the process.

Looking at the second state of the dual structure, in order to enable business performers to bring changes to the corporate structure, they must play active roles by suggesting improvements on tasks defined at a low level of granularity, which is the manageable level of tasks corresponding to their work. Some works have emphasized the importance of involving users in process modeling [20]. But they had not yet realized how users can give more consistent contributions for business process improvement using their own knowledge, without having to acquire new skills (i.e. business analysis) before they are able to contribute.

In order to enable the active participation of business performers (a.k.a. end-users), the UI-Business Alignment methodology advocates different approaches, where different professionals can give their own contribution, depending on their skills and goals. These stakeholders can mainly contribute in the following manner:

- *Business analysts* can identify prospective improvements on business processes to address new corporate strategies;
- *UI designers* can identify improvements directly on the UI design to address specific interaction needs;
- *Usability experts* can identify better user interactive tasks that impact business operations;
- *Operation managers* can identify improvements on the performance of processes to address market opportunities that impact the processes and systems as a whole;
- *Users* can give suggestions towards improving user interaction and their productivity to be considered in the business context.

Each of these stakeholders contribute more precisely on specific approaches. The forward approach starts with business processes and stimulates the participation of operation managers and business analysts. The middle-out starts with interactive tasks and stimulates the participation of usability experts and UI designers. The backward approach starts with UIs and stimulates the participation of basically anyone in the organization, more precisely, UI designers and end-users.

UI-Business Alignment Methodology: The methodology proposes a strategy to map the core models related to enterprise systems and build a network of links that supports traceability and impact analysis when changes are requested in any of these models. The core actions to create the network of links can be applied in two main approaches: forward and backward:

Forward Approach simulates what could be the future user interaction to execute the process in situations where there is a process and there is no system.

Backward Approach uses navigation patterns in the system as a source of data to exploit business processes in situations where there is a system and there is no process.

The methodology is effectively adopted by starting with a critical process or system, which could be a great source of data to be analyzed in an initial project. After applying the methodology at the first time, a procedure is prepared to spread the

methodology through other critical and non-critical processes and systems considering the organizational context in other projects.

After the mappings are created, the models are managed through the execution of rules in three main approaches, as depicted in Fig. 3.

The **forward approach** starts with changes made on BP models, which can be done from a variety of reasons, including new or alternative ways of doing things, new business opportunities, organizational changes, new regulations; etc. When changes are made, rules are executed from the business process to the task model, persisting the changes on the task model, even if the task model does not yet exist, thus deriving a task model from the business process. Then, the rules are executed from the task model to the UI model, persisting the changes on the latter, assuming the UI model has been created beforehand.

The **middle-out approach** starts with changes made on task models, which can be done to perform new tasks that improve the user experience. In this approach, the rules are executed from the task model to the business process, persisting the changes on the latter, even if the business process model does not yet exist, thus deriving a business process from the task model. Then, the rules are executed from the task model to the UI model, persisting the changes on the latter, assuming the UI model has been created beforehand.

The **backward approach** starts with changes on UIs because of defects to be fixed, better user understanding of the systems' features, new technology, etc. In this case, rules are executed from the UI model to the task model, persisting the changes on the task model, assuming the task model has been created beforehand. Then, the rules are executed from the task model to the business process, persisting the changes on the latter, even if the business process model does not yet exist, thus deriving a business process from the task model.

There are no rules specified from the UI model to transform to the actual UIs of systems. There is a synchronization to indicate what has to be updated in the UIs directly by UI designers. This methodology considers that UI designers are relevant to create UIs for enterprise systems considering their human interpretation over user

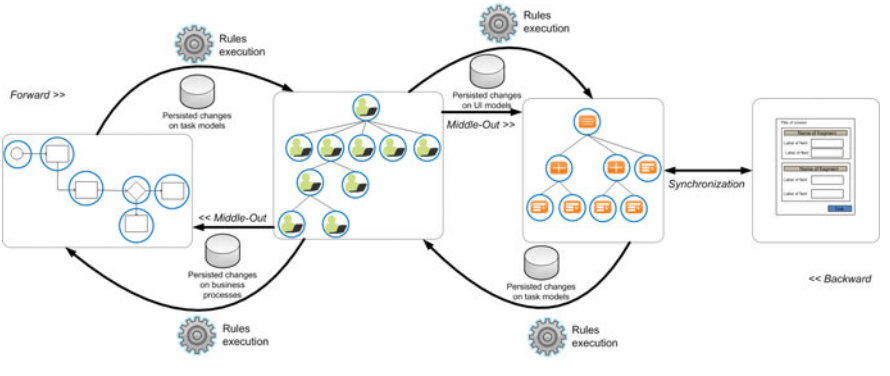

Fig. 3 UI-Business Alignment applying rules in different approaches

experience, user needs, and design ideas. The use of rules is a primordial aspect in this methodology because they enable tracing one element in the target model back to its causing element in the source model and provide transformations from source to target and vice-versa. But this tracing or transformation is useful only when people who perform the processes are active in analyzing and identifying bottlenecks in the business processes and user interfaces that support these processes.

For that reason, the backward approach is primordial to enable the participation of business performers. In this approach, end-users give suggestions based on their own experience with the systems. This solution is independent of how processes are designed and systems are developed. In most solutions in the market, business processes must be executed in specific engines and the system must be developed in a specific technology (e.g. web services) to allow collecting data as the system is used. Our solution is applicable in a wider range of corporate scenarios since we can monitor any kind of system that are representations of certain business processes (e.g. an order management system for provisioning products). In this approach, stakeholders do not need specific skills in order to be able to delineate which part of the system brings impediments for the successful completion of their goals. Therefore, we present how business performers can be active participants and the application of the rules aiming for better user interaction and improved process by presenting details of the model-driven perspective in the backward approach.

3 Model-Driven Perspective

The main models used in the UI-Business Alignment methodology are: Business Process Model, Task Model, User Interface Model. The UI-Business Alignment methodology is founded on Business Process Modeling Notation (BPMN) [21] for business process models and on the Cameleon Reference Framework and, more precisely on UsiXML for the definition of the task and UI models.

Business Process Model: This methodology is founded on BPMN that provides the standard for business process modeling. However, there are other business process notations that could be used with the UI-Business Alignment methodology. The main requirement to apply a business process notation in this methodology is that it is able to decompose activities in smaller parts, such as a sub-process in BPMN that can be decomposed in as many levels as needed to specify a complex activity.

Task Model: The types of UI Tasks in the task model are: (i) abstract, grouping of related tasks; (ii) user, task performed manually by a human being with no interaction with a system; (iii) interactive, task performed by the user interacting with the system; and (iv) system, task executed by the system itself. For the purpose of mapping business process activities that are to be performed by users when interacting with enterprise systems, we have delimited the scope to consider only interactive tasks in the traceability chain. Because we deal only with UIs, we do not need manual (user) tasks and automatic (system) tasks. Manual and automatic tasks by themselves do not involve the direct user interaction with a system. Thus, we have

restricted our needs to interactive tasks only. Consequently, we simplify the task model with only one kind of task, which also contributes to increase the task model acceptance by providing simplicity.

User Interface Model: UI is a means by which users interact with the system. One of the main challenges of models aiming to formalize user interaction is to follow the fast technological evolution of devices and new kinds of interactions. This problematic has been our major motivation to investigate which model is the most appropriate to cover so many possibilities. In a survey comparing existing UI modeling languages [22], the abstract UI model seems to be the one that covers the broadest possible range of targets for the broadest possible range of delivery contexts. Therefore, we came up with an extension of the abstract UI model based on UsiXML. The difference is that the terms are not used literally because this UI model, for the purpose of UI-Business alignment, needs easy names and less UI components for simplicity. This simplification of the UI model aims to make it easily understandable by end-users when delineating the UI model. This different representation of the UI model does not substitute the language, which is used to guarantee interoperability. The UI model structure is comprised of UI components, which are components used for user interaction that start in an atomic level and are further composed in broader levels, namely: screen group, screen, screen fragment and screen element.

The core models have been structured with a set of elements that are primordial to be linked among them with the primary goal of UI-Business Alignment. In order to facilitate modifying how the models are structured, depending on the strategies adopted by the organization, an expert system approach based on production rules has been selected. This approach provides flexibility since it enables updating the rules whenever there is a change on how the models are structured and managed, without the need for direct maintenance of the platform that supports the methodology. This approach makes this framework adaptable to different organizational contexts by accepting that organizations may adapt their strategies to model processes and design user interfaces.

To define the knowledge of an expert system, it is necessary to specify the objects that are important to the domain, the properties of these objects and the relationships among them, and assertions about these objects, composing a *working memory* that is constantly changing during the operation of the system [16]. The format of our working memory is a graph with links among the model elements, forming the *network of links*.

Building the network of links is the process of mapping the elements of the core models of our methodology. These mapped elements form the working memory elements, called *objects*, that are instances of these model elements. These objects are of different types, represented by the *entities* of the models, such as BP activities, UI tasks, UI components.

For each entity of the models, there is a set of rules that verifies the integrity of the models when an event happens on any object of that entity. An *event* is any action that changes the state of the object, such as create, update, delete, and other types

of operations performed in the models. Performing an event on any given object triggers the execution of rules for that object. Thus, when an event is done in any of the objects, a set of rules is executed in order to generate an impact report that lists all the elements on these models that have been impacted by the changes and the proposed changes to maintain consistency.

As a contribution for this research, we have defined rules that adhere to the basic properties of transformation rules [23] and follow the general format of production rules [16]. The rules have been written using the Drools Rule Language (DRL) because it is easy to understand, flexible, reusable, has a reasonable performance, independent lifecycle, can be easily embedded into existing applications, among other advantages of its declarative style solution [24].

To make the rules understandable by a larger range of stakeholders involved in managing the rules, such as non-technical stakeholders, it is productive to adopt human-readable rules using *Domain Specific Language* (DSL) [23]. A DSL is a computer language that is focused on a clearly definable problem domain, rather than a general purpose language that is aimed at any kind of software problem. A DSL file specifies the translation of sentences from the problem-specific terminology into rules. In Drools, a DSL file maps DSLR sentence (human-readable rule) to a DRL rule (rule in the Drools rule language). DSLR is useful to demonstrate an important reason for using rules; which is that rules can be customized by different stakeholders. Therefore, stakeholders can update and create new rules according to their approaches adopted for BP modeling, user experience, UI design, among other aspects.

To execute the rules, it is necessary to verify if the object matches specific conditions. For that, there are three main conditions that are checked, which are: if the object has been recently created, updated or deleted. The object is understood as recently created when its id is empty, since the id is generated automatically by the database only when the object is permanently persisted. The object is understood as updated when the id is different from empty, which means that the object already exists with a valid id. The object is understood as deleted when the reference to it is null, which means that the object does not exist anymore.

The Drools rule definition starts with the rule name; the condition and consequence sections follow. Drools keywords are *rule*, *when*, *then*, and *end*.

```
rule ''name of rule''
when
    conditions
then
    consequence;
end
```

The condition defines the patterns that the rule matches with. The consequence is a block of code that is executed when all of the patterns within the condition are matched. The rule condition can declare variables of an object type. The variable name starts with a $ symbol, it can be declared up front to be used later in the rule. The condition verifies if the object field matches a certain value, which can be of

different types, such as predefined variable, string, regular expression, date, boolean and more. A condition can have multiple field constraints joined by additional keywords, such as an implicit *and* (i.e. a comma), *or*, *not*, *exists*, etc.

$$\$variable : Object\ (field_i == value_i, field_j == value_j)$$

In the rule consequence, Drools has convenience methods to be executed as simple actions: `modify`: to update existing facts in the working memory; `insert` to insert new facts in the working memory; `retract` to remove existing facts in the working memory.

Considering that each entity of the models has a set of rules that verifies the integrity of the models when an event happens on any object, we add the rule set name in the beginning of each rule definition. The rule set has the name of the entity to which it is related. The name of the rule follows this format: `impact on <<element>> when <<operation>> <<element>>`.

An example of a rule in the Drools rule language is presented as follows, which verifies when there is an element of type `activity` that has just been created (i.e. with a field id that is empty), then add to the working memory an element of type `UITask` related to this activity.

```
ruleset ‘‘BP Activity’’
rule ‘‘impact on UITask when adding BP Activity’’
when
    $activity: Activity(id == "")
then
    insert(new UITask($activity));
end
```

The execution of these rules follows a cycle of three steps [16] that uses the rules from a rule catalog, as follows:

- *recognize*: find the rules whose conditions are satisfied in the current working memory (e.g. rules for a BP activity);
- *resolve conflict*: among these rules recognized in the first step, choose the one that should be executed (e.g. rules that process the addition of a new BP activity that is not yet related to any UI task);
- *act*: perform the consequent actions of the rules selected in the second step to change the working memory (e.g. add the new UI task for this added BP activity).

This cycle continues until the conditions tested in the rules are false, which means that the models are supposed to be consistent at this point.

The rules that have been defined as pertaining to a rule catalog are explained and exemplified using a scenario from a case study at the telecommunications industry. The execution of the rules and its results are presented applying the backward approach.

Context of case study: This case study occurred in a customer-unit responsible for Business-to-Business transactions with twenty-three professionals, among

managers, business analysts and IT professionals, who participated in meetings and interviews during four months.

The context under analysis is the provisioning of the Integrated Telephony Solution (ITS) products of the enterprise Voice over IP (eVoIP) solution for large enterprises that want to integrate their data and voice traffics on the same data network. From this context, we consider one category of these products that are determined by the quantity of dialing numbers (number of the physical phones) that the customer requires. Each dialing number may be associated to a package that represents voice service packages, such as call center, auto-attendant, receptionist, etc.

This case study is suitable to illustrate the backward approach since it did not have the business process modeled. For that reason, we start with UI model derivation from the UIs of the available system, followed by task model derivation and business process definition.

3.1 User Interface Model Derivation

A business analyst explained how the provisioning of ITS products works in practice and a system analyst presented how the provisioning is done using the system. After understanding the system, we observed 8 users interacting with the ordering system to provision eVoIP products.

During the observations, users mentioned that the selection of service packages was one aspect of the provisioning of products that was slowing them down. First, they selected voice service packages and then, they had to check if the packages remained the same after the attribution of dialing numbers. But since the attribution of numbers is an automatic process, this checking by users is unnecessary since they cannot interfere on those numbers, just on the packages that they have themselves selected previously.

Fig. 4 depicts exactly the two screens in which users first select service packages for eVoIP products that have not yet received the dialing numbers (i.e. Negotiate features). Once these products have the numbers, users can see the selected packages for each dialing number (i.e. Update features). Each screen uses a different way to display this information: the first one uses a tree and the second one uses a matrix.

This issue identified by end-users themselves decreases their productivity when managing the orders of customers. End-users proposed to merge these two UIs in one screen that is more efficient. Merging these UIs is done by selecting the most relevant UI components.

Based on the existing UIs of the ordering system, we have specified the UI model for the UIs. Fig. 5 depicts the UI model for the screen group to provide product, composed of several screens, but here the focus is on these two screens where users can select and check voice service packages. These screens have some screen elements in common (marked with dashed lines).

When two screens are merged, what happens is that a new screen is created with existing content. This change triggers a set of rules that supports deriving new UI tasks and updating the task model.

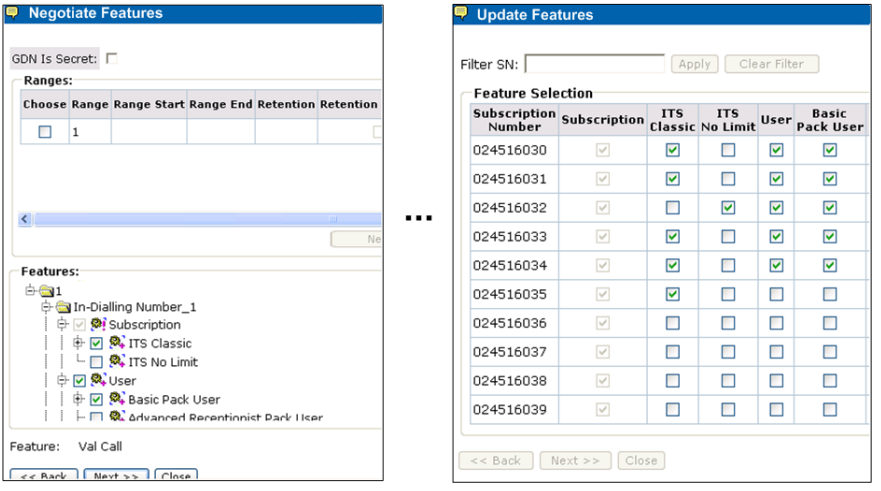

Fig. 4 Screens to select and check voice service packages

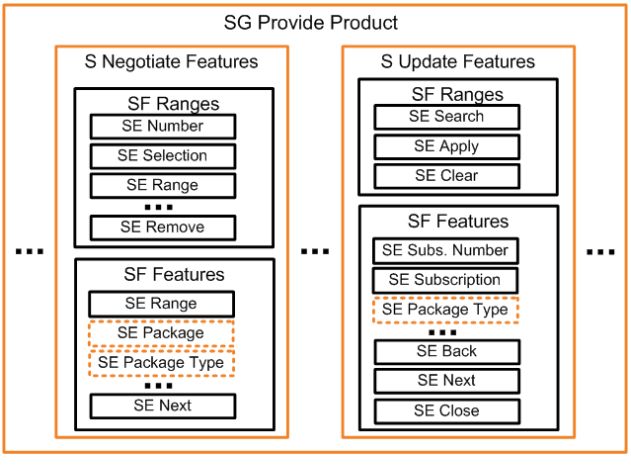

Fig. 5 UI model for the screens to select and check voice service packages

3.2 Task Model Derivation

Once the UIs were understood and the UI models created, the UI tasks in task models were derived from the execution of the rules. An extract of the task model to introduce products is depicted in Fig. 6.

Once this task model is created, whenever changes are made on its related UI model, this task model needs to updated accordingly.

For two screens that are merged, the working memory receives an instance of the new screen, of its pre-existing screen fragments and the UI tasks related to these

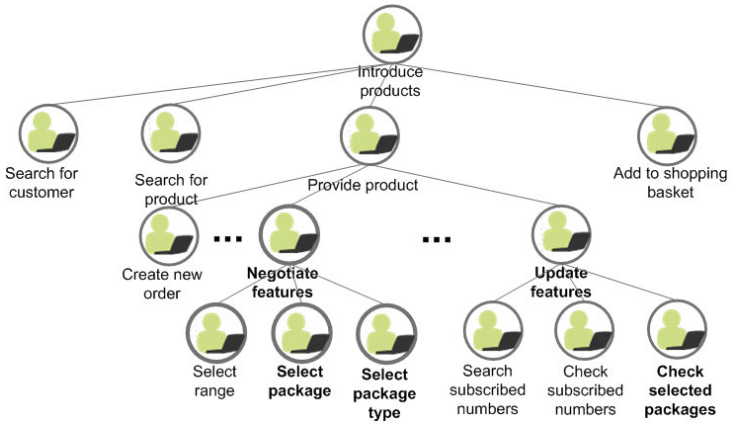

Fig. 6 Task model related to introducing products

screen fragments. The conditions that are verified in this case are: if there is a new screen with screen fragments, and this screen is not yet related to any UI Task, but its screen fragments are already related to UITasks. If this is the case, the task model needs to be updated. A set of rules related to Screen is triggered and the rule that positively addresses this condition is executed. The execution of the following rule creates a new UI task for this screen and creates sub-tasks for this UITask using the existing sub-tasks related to its screen fragments:

```
ruleset ''Screen''
rule ''impact on UI Task when merging Screens''
when
     The Screen has an empty id and does not have related
                UITasks
     The ScreenFragments have related UITasks
then
    create UITask for the Screen;
    modify this UITask with the association with
           sub-tasks related to the ScreenFragment;
end
```

Fig. 7 shows that a new screen is created with existing screen fragments from the two repeated screens. Since only the existing screen fragments are related to UI tasks, the rule results in the creation of a new UI task and associating it with the existing sub-tasks.

To delete the screens that have been merged, since a new screen is already substituting them, the working memory receives the instance of the deleted screen and of the UI tasks related to it. A set of rules related to screens are triggered and the rules that positively address the condition that verifies if a screen with related

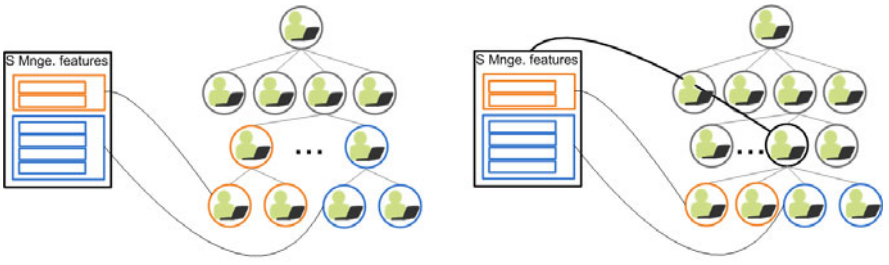

Fig. 7 UI model for the screens to select and check voice service packages

UI tasks was deleted is executed, thus deleting the related UI tasks through the following rule:

```
ruleset ''Screen''
rule ''impact on UI Task when deleting Screen''
when
    The Screen is null and
    There is an existing UITask
then
    remove UITask;
end
```

3.3 Business Process Definition

With the details of UI tasks, it was possible to specify the high level activities of the business process to provide product, depicted in Fig. 8. This business process model can be further refined by business analysts, checking accordance to business rules and strategies. This refinement is done depending on the levels of granularity adopted to model business processes: From an operational perspective, the activities are defined until the smallest action. From a process perspective, the activities specify what is performed and not details of how it is performed [25].

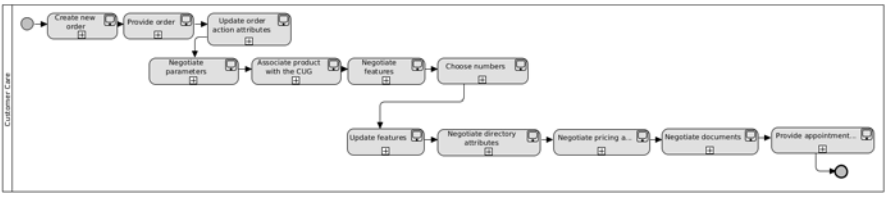

Fig. 8 Business process model to provide product

For each added UI task in the task model, the following rule was also executed to add the equivalent BP activities:

```
ruleset ‘‘UITask’’
rule ‘‘impact on Sub Process when adding UI Task’’
when
    The UITask has an empty id and does not have
                related activity
then
    create activity for UITask;
end
```

Once the business process is created, whenever changes are made on its related task model, this business process model needs to updated accordingly. Considering that some tasks were deleted because two screens were merged, some activities in the business process model need to be deleted. When UI tasks have been deleted, the working memory receives the instance of the deleted UI tasks and activities related to it. A set of rules related to UI tasks is triggered and the rules that positively address the condition that verifies if a UI task with related activities was deleted is executed, thus deleting the related activities through the following rule:

```
ruleset ‘‘UI Task’’
rule ‘‘impact on Activity when deleting UI Task ’’
when
    The UI Task is null and
    There is an existing Activity
then
    remove Activity;
end
```

This example has illustrated how the models are created and managed. Initially, the associations between business process models, task models and UIs is created, which builds a network of links. With this network built, it is possible to navigate in the network in any direction, which aids in identifying the impact of changes, supported by the execution of rules and updating the models according to the changes. This example particularly shows the analysis of the network of links starting with the UIs through suggestions of business performers taking into consideration their knowledge about the domain and expertise with the system.

4 Discussion

For the methodology to be efficiently applied, it is necessary to have a software platform to support the execution of these rules. The behavior of the platform is based on the methodology; accordingly, this platform can be used by different stakeholders, depending on their goal within the organization. To support the creation of the network of links, each stakeholder has a specific role.

Business analyst - When business analysts either create a new business process or update an existing one, they import the business process model into the platform

as a starting point to create the network in the forward approach. Once this is done, the task model is automatically derived based on the imported business process.

Usability expert - Once a first version of the task model has been automatically created, most of the repetitious work is done and usability experts can focus on reviewing it and adding UI tasks that represent the user interaction.

UI designer - As the usability expert is working on the task model, the UI designer lists: the systems used to perform this business process, the users of these systems, and the UI components of each system. The level of detail given to the list of UI components depends on the level of detail expected for the result of the impact analysis. Once all these elements are listed and the usability expert finished updating the task model, the UI designer can associate the UI tasks with the listed UI components.

Operation Manager - When the network of links is created, the operation manager can request an impact analysis concerning a specific change that he/she prospects. When the operation manager, views the result of the impact analysis with the list of suggested changes, he/she can make a solid decision.

Each of these stakeholders is involved in managing specific models. We assume that the models created are correct, consistent and complete. They are complete considering that these models are designed in an iterative manner. Each iteration produces a version that is complete for the current context, but they are still open for changes and improvements. The platform is aimed to support stakeholders with information to help them maintain consistency between the models. However, the platform does not aim to indicate that a model is right or wrong because this is pertaining to the human abstraction skill. On the other hand, the platform indicates if a model is consistent or inconsistent, according to the conventions described in the rules. Being wrong may mean that it is different from reality or that it has a deadlock in the process flow, as an example of an error that stakeholders may make when modeling. As classified in [26], stakeholders normally fall in one of the seven sins of specification, namely: noise (i.e. irrelevant information), silence (i.e. lack of significant information), over-specification (i.e. specification of features out of scope), contradiction (i.e. incompatible definitions of a same feature), ambiguity (i.e. possibility to interpret a definition in different ways), forward reference (i.e. mention information before it is defined) and wishful thinking (i.e. lack of realistic aspects to validate the feature specified). Therefore, it is an assumption that stakeholders are confident in creating the models and that they at least try to maximize the correctness, consistency and completeness of the models.

The rules have been defined to provide flexibility to the platform since they can be updated whenever there is a change on how the models are managed, without the need for direct maintenance of the platform. The platform provides functionalities to make transformations from the business process into the task model, and backwards, avoiding creating models from scratch. The platform helps stakeholders to update UI tasks, delineate UI models and create the mappings and the rules are at the foundation of that support as they help maintain consistency between the models.

Besides the platform that can add productivity in the application of the methodology, there is also the return on investment that the methodology can bring to

specific projects. For each project, the methodology can be evaluated to check that the benefits of applying it outgrow its costs, thus proving its feasibility in a corporate setting and motivating stakeholders to continue applying it. Besides the profit that the methodology might bring from resulting improvements on business processes, the human perspective in this methodology makes end-users become active agents in the company. The methodology enables a free channel where they can give their opinion about user interaction. This opinion is translated into business processes so operation managers can ponder on their impact. When end-users' suggestions are put into practice and these changes improve their work, they are motivated to collaborate. The more an end-user's suggestions helps other end-users, the stronger is their reputation within the group (e.g. customer representative department) and higher is the trust between people.

5 Conclusion

This methodology is aimed for enterprise systems in organizations with extensive business processes and hundreds of users using thousand of user interfaces. The main goal is to keep enterprise systems aligned with business processes from the point of view of business process performers and business process designers, offering a complementary approach to existing solutions of IT-Business alignment. These complementary solutions are more concerned with requirements and implementations, while we focus on UIs, the part effectively in use by business performers (who are also end-users).

Because of this concern with business performers, this methodology becomes human-centered since it is concerned with the people who perform the business processes and use enterprise systems; more than merely on technological or methodological aspects, supported by the theory of structuration [18].

To demonstrate the effectiveness of the business process perfomer-centred approach, the methodology has been quantitatively analyzed using sensitivity analysis [27] to evaluate the cost-benefit analysis of its application in a telecommunications company. This quantitative analysis has demonstrated that applying it brings up to 60% of return on investment related to process improvement and user experience.

To further develop the human-centered aspects of this methodology, the authors are analyzing techniques to be used by business performers to aid them in giving suggestions about the user interaction and how they collaborate with others to achieve this goal. Another important aspect to be analyzed is how well business process models are understood by different stakeholders, how data is handled in the execution of business processes, etc. In the current state of the methodology, it does not control the business process in execution time, especially since it is risky to execute the rules and change the models at runtime. But for guidance and visualization purposes, a solution for stakeholders to be aware of the state of the process in execution time would be to change corporate applications to store information when users interact with specific UIs to know where they are in the process by looking at

the mappings. Consequently, there are other requirements that can be considered as an extension of this methodology.

Overall, this research has achieved good results both in the academia and in the industry and it is differentiated by considering a human perspective for UI-Business Alignment, which had not yet been addressed in the literature neither in the industry. Its characteristic of being multi-disciplinary and complementary to existing IT-Business alignment strategies opens new themes for collaboration with researchers and practitioners from other communities.

Acknowledgements. We acknowledge the support of the ITEA2 Call3 UsiXML project (under reference 20080026).

References

1. Janssen, C., Weisbecker, A., Ziegler, J.: Generating User Interfaces from Data Models and Dialogue Net Specifications. In: Proc. of InterCHI 1993, pp. 418–423. ACM Press, Amsterdam (1993)
2. Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M.A.: Beyond Data Models for Automated UI Generation. In: Proc. of HCI 1994, pp. 353–366. Cambridge Univ. Press, Cambridge (1994)
3. Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 16–31. Springer, Heidelberg (2005)
4. Mori, G., Paterno, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Transactions on Software Engineering 28, 797–813 (2002)
5. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1998)
6. Davenport, T.: Process Innovation: Reengineering work through information technology. Harvard Business School Press, Boston (1993)
7. Henry, P.: Process-User Interface Alignment: New Value. From a New Level of Alignment. Align Journal (2007)
8. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1152–1163. Springer, Heidelberg (2008)
9. Sousa, K., Mendonça, H., Vanderdonckt, J.: A Model-Driven Approach to Align Business Processes with User Interfaces. International Journal of Universal Computer Science, Special issue on Human-Computer Interaction 14, 3236–3249 (2008)
10. Bodart, F., Hennebert, A.M., Leheureux, J.M., Vanderdonckt, J.: Computer-Aided Window Identification in Trident. In: Nordbyn, K., Helmersen, P.H., Gilmore, D.J., Arnesen, S.A. (eds.) Proc. of 5th IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact 1995, pp. 331–336. Chapman & Hall, London (1995)
11. Paterno, F.: Task Models in Interactive Software Systems. In: Chang, S.K. (ed.) Handbook of Software Engineering & Knowledge Engineering. World Scientific Publishing Co., Singapore (2001)
12. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 289–308 (2003)

13. Teo, L., John, B.E.: Towards Predicting User Interaction with CogTool-Explorer. In: The Human Factors and Ergonomics Society, 52nd Annual Meeting, New York (2008)
14. Traetteberg, H.: UI Design without a Task Modeling Language Using BPMN and Diamodl for Task Modeling and Dialog Design. In: Forbrig, P., Paternò, F. (eds.) HCSE/TAMODIA 2008. LNCS, vol. 5247, pp. 110–117. Springer, Heidelberg (2008)
15. Sukaviriya, S., Mani, S., Sinha, V.: Reflection of a Year Long Model-Driven Business and UI Modeling Development Project. In: INTERACT, vol. 2, pp. 749–762 (2009)
16. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier, San Francisco (2004)
17. Stolze, M., Riand, P., Wallace, M., Heath, T.: Agile development of workflow applications with interpreted task models. In: Winckler, M., Johnson, H. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 2–14. Springer, Heidelberg (2007)
18. Giddens, A.: The Construction of Society. Political Press, London (1984)
19. Zacarias, M., Caetano, A., Magalhaes, R., Sofia Pinto, H., Tribolet, J.: Adding a Human Perspective to Enterprise Architectures. In: DEXA - WEISE 2007 - International Workshop on Enterprise Information Systems Engineering, pp. 840–844 (2007)
20. Stoitsev, T., Scheidl, S., Flentge, F., Muhlhauser, M.: From Personal Task Management to End-User Driven Business Process Modeling. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 84–99. Springer, Heidelberg (2008)
21. OMG, Business Process Modeling Notation Specification, v. 1.2. (2009), `http://www.omg.org/spec/BPMN` (accessed September 10, 2010)
22. Souchon, N., Vanderdonckt, J.: A Review of XML-Compliant User Interface Description Languages. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 377–391. Springer, Heidelberg (2003)
23. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley, Reading (2008)
24. Bali, M.: Drools JBoss Rules 5.0 - Developer's Guide. Packt Publishing, Birmingham (2009)
25. Bhattacharya, K., Caswell, N., Kumaran, S., Nigam, A., Wu, F.: Artifact-centric Operational Modeling: Lessons learned from customer engagements. IBM Systems Journal 46, 703–721 (2007)
26. Meyer, B.: On Formalism in Specifications. IEEE Software 3, 6–25 (1985)
27. Krajewski, L., Ritzman, L., Malhotra, M.: Operations Management Processes and Value Chains, 8th edn. Pearson International Edition, NJ (2007)