Towards a Method to Generate GUI Prototypes from BPMN

Eduardo Diaz *Jose Ignacio Panach* *Silvia Rueda*
Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica
Universitat de València
Avenida de la Universidad, s/n, 46100, Burjassot, València, Spain
diazsua@alumni.uv.es, {joigpana,silvia.rueda}@uv.es

Oscar Pastor
Centro de Investigación en Mètodos de Producción de Software,
Universitat Politècnica de València
Camino de Vera s/n, 46022, Valencia, Spain
opastor@pros.upv.es

Abstract— Business Process Model and Notation (BPMN) provides organizations with a standard that facilitates further compression of the business process. BPMN focuses on the functional processes, leaving the development of interfaces to one side. Thereby, interface design usually depends on the subjective experience of the analyst. This article aims to propose a new method to generate user interfaces from BPMN models and Class Diagrams. The proposed method is based on the identification of different rules and makes use of stereotypes to extend BPMN notation. The rules have been extracted from seven existing projects on the Bizagi repository. Specifically, the proposal is based on the extraction of rules for generating user interfaces based on three widely used patterns: sequence pattern, exclusive decision pattern and synchronization pattern. As a result of our proposal, eleven new stereotypes have been added to BPMN notation. These stereotypes allow generating interfaces based on business process models. For a better understanding, the proposed stereotypes have been applied to an illustrative example. The results show that this work is a "step forward to" the automatic code generation from models.

Keywords—BPMN; method; rules; stereotypes; user interfaces

I. INTRODUCTION

Business Process Model Notation (BPMN) provides organizations with the ability to understand their internal business procedures in a graphical notation, as well as the ability to communicate these procedures in a standard way. The basic conceptual primitives in BPMN models are events, tasks, gateway, swimlane and flows [1]. Currently, BPMN is required to document business processes with the aim of reducing time and costs by improving processes [2]. Existing BPMN modelers allow to design diagrams easily, providing options to improve business processes. Among other modelers, we can highlight: Bizagi [3], Adonis NP [4] and Auraportal [5]. In all the existing modelers, the interface design is put in the background, leaving this task to the designer, who must make an effort of implementing interfaces according to the BPMN model. How the interfaces are currently derived from BPMN models in a hand-made way, without any kind of procedure, only depending on the analyst experience. This means that despite the effort made building the BPMN, it is not useful when designing the interfaces at the end. In addition, normally, analysts that build the BPMN models are not the same designers who implement the user interface, generating a gap between what is described in the BPMN models and what it is really implemented in the interface.

In this paper, we propose a method to generate interfaces from BPMN models. The process has been summarized in Fig. 1. The approach is based on the study of three widely used patterns in business process models (Sequence Pattern, Exclusive Decision Pattern and Synchronization Pattern). We have studied the use of these three patterns in 7 Bizagi projects [6] to extract common transformation rules from BPMN models to code. Bizagi repository has both the BPMN models and the implementation of the interfaces, so we can analyze the mapping between both elements. Analyzing how each BPMN model results in an interface, we can extract transformation rules that can be generalized for any project. When the rules have several alternatives to generate the interfaces, we need an unambiguous semantic to specify which alternative will be used. For this aim, we have extended the BPMN model with new stereotypes that allow to specify how to generate interfaces in those rules with more than one alternative. The stereotypes of the BPMN model have been also complemented with stereotypes of UML Class Diagram, since part of the interface depends on persistency model.

The approach has been applied in an illustrative example to show how we can work with extended BPMN models along with stereotypes and complemented with the attributes of the Class Diagram. The example also clearly shows the correspondence between the BPMN models and the end user interface.

We have also conducted an early validation applied in other 7 projects different from Bizagi. For each pattern and its associate transformation rules, we have analyzed: what portion of the BPMN model is covered through the three patterns used in our approach; what transformation rules could be used in the projects; and the agreement between the real interface that implements the projects and the interface that would be generated with our proposed rules.

Results of the validation show that our 3 patterns cover most of the BPMN models, being the sequence pattern the most used. The validation has been also useful to identify deficiencies in our approach. We have identified that there is a short percentage of the interfaces that cannot be generated with our rules. We need to analyze more patterns to derive more transformation rules in order to get a holistic transformation process from BPMN models to interfaces.

The remainder of the paper is structured as follows: Section 2 reviews the literature related with the generation of user interfaces and the use of BPMN patterns. Section 3 presents BPMN patterns. Section 4 defines the rules to generate interfaces from BPMN patterns. Section 5 presents rules and stereotypes to extend the BPMN notation. Section 6 shows an illustrative example. Section 7 shows an early validation. Finally, Section 8 concludes the paper.

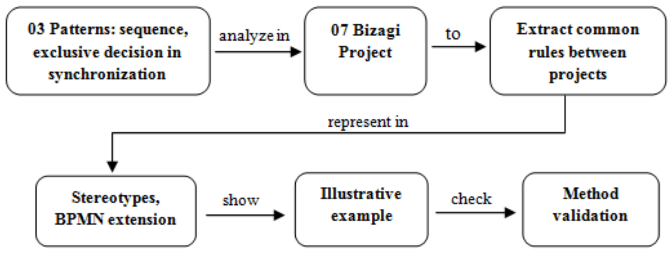

03 Patterns: sequence, exclusive decision in synchronization
analyze in
07 Bizagi Project
to
Extract common rules between projects
represent in
Stereotypes, BPMN extension
show
Illustrative example
check
Method validation

Fig. 1. Content of this work.

II. State of the Art

This section reviews previous works of the literature related with our proposal: the user interface generation from BPMN patterns. These works are classified into two groups: (1) Generation of user interfaces from BPMN, which deals with methods or techniques to generate software components; (2) Patterns analysis in BPMN models, which deals with patterns to work with BPMN.

Generation of user interfaces from BPMN

Lei Han et al. [7] defined an approach of derivation of user interfaces from BPMN models. That approach is based on a role-enriched business process model developed with tasks descriptions and associated data. The model is specified in an extended BPMN. A set of control flow patterns and data flow patterns are identified for the UI derivation. A comprehensive set of constraints and recommendations are specified for supporting the UI generation and updating. The main problem with that approach is that currently there are only generation rules to generate a model to represent abstract interfaces. There are no rules to generate final interfaces yet.

Kenia Sousa et al. [8] defined a model driven approach for organizational engineering in which user interfaces are derived from business processes. The approach is based on the Cameleon Reference Framework, which consists of four steps: (1) business process modeling in the context of organizational engineering; (2) task model derivation from the business process model; (3) task refinement; and (4) user interface model derivation from the task model. Each step contributes to specify and refine mappings between the source and the target model such a way each model modification can be adequately propagated to the rest of the chain. The work of Kenia Sousa et al. focuses on using an external framework to transform business processes model to task model, and next designing user interfaces. The main problem with this approach is that it does not propose any method or technique to perform transformations.

Chun Ouyang et al. [9] defined transformation rules between BPMN models and BPEL definitions. This approach includes an integrated set of techniques to translate the captured models using a central subset of patterns BPMN to generate BPEL code (Business Process Execution Language for web services). This technique transforms business process diagrams into mapped BPEL blocks. The approach is based on the semantics of components in terms of Petri nets, whose properties are analyzed and transformed into BPEL code. The main problem is that currently the transformation techniques of BPMN models for Business Process Execution Language for web services (BPEL) translate from non-structured graphs to similar textual languages. This transformation only works in non-complex BPMN models.

Wided Bouchelligua et al. [10] defined a user interface modeling based on workflows. The approach follows a set of model transformations according to a Model Driven Engineering (MDE) approach. First transformation is the proposition of a model driven approach to derive a plastic UIs from a workflow. Second transformation lies in the use of Business Process Modelling Notation (BPMN) for the modelling of the interaction models (Task Model, Abstract User Interface and Concrete User Interface). In that work, the main problem is that they do not describe rules to show user interfaces from the BPMN models. The figures presented by the authors do not clearly explain what they represent.

Javier Gonzales et al. [11] defined an approach to transform BPMN models to software artefacts. The approach consists in three steps: (1) refining BPMN models through re-engineering; (2) automating patterns to derive software analysis; (3) designing artifacts (UML Class Diagrams or UML Use Cases). These artifacts will be later on used to develop the software components that support the business process activities. The main problem of that work is that it focuses on the first two steps of the approach, where there is not still interfaces generation.

As summary, all the studied articles are developing approaches for the generation of user interfaces that implement BPMN models. The generation rules identified in previous existing works are not suitable for most real BPMN models since most of these works are in an initial stage where transformation rules have not been defined yet.

Patterns Analysis of BPMN

Next, we describe some approaches that work with patterns in the construction of BPMN models.

M. Kabir et al. [12] defined reusable process patterns for enhancing the business process model. The approach includes the definition of a system to classify and map process patterns to BPMN models. The work focuses on the use of process patterns to extend the BPMN model with new characteristics, taking into account a set of evaluation criteria. The main problem is that it does not focus yet on the hierarchy of process patterns, that is, a process pattern that contains more process patterns. This limitation in the hierarchy involves that the approach could not be used in complex BPMN models.

Anis Boubaker et al. [13] defined a systematic method for extracting a value chain (business model) from a process model in BPMN through patterns. The approach consists in the application of patterns extracted from the analysis of 20 business projects. These extracted patterns are structural and conceptual patterns, conversion patterns, exchange patterns, subcontracting patterns, insertion patterns and rental patterns. Each pattern provides a short textual description along with a visual and formal representation, mapping common concepts between BPMN and Resources Events and Agents (REA). The main problem with this method is that only the REA events are extracted from some BPMN models. Functionality, persistence or interface design details cannot be extracted from REA.

Alaaeddine Yousfin et al. [14] proposed the improvement of business processes through variability patterns. There are two types of variability patterns: product patterns, which act on the performance metrics of the improved business process; and process patterns, which make the steps for building a variable business process free from confusion and ambiguity. This allows to extend the BPMN model by defining new events. Each pattern is shown in an illustrative example for technical validation. The work of Yousfin focuses on patterns of process variability. Variability divides the tasks according to specific variables proposed by the authors. Next, these tasks can be subdivided according to business rules, resulting in a BPMN extension. The main difference with our proposal is that we use behavior patterns of BPMN but not process patterns.

As summary, all these articles propose a set of patterns for BPMN models through rules identified in other projects. These patterns have been identified with the aim of enhancing BPMN models. For these proposed patterns, a validation process within different projects is needed in order to have more reliability in their use. Note that most of the existing research in patterns has not been evaluated empirically.

III. BPMN Patterns

BPMN is a language where patterns are frequently used. The BPMN patterns are situations that commonly happen in any business process model [15].

These patterns help reuse solutions to face problems suffered in any business process, so they are frequently used. This is the reason why we chose the interface generation from such frequently used patterns. This way we can ensure that in any BPMN model we can generate some interfaces (at least the part of the models based on patterns). This work focuses on 3 patterns: Sequence Pattern; Exclusive Decision Pattern and Synchronization Pattern. This choice is justified since all of them are the most frequently used in different BPMN models.

Sequence pattern, this pattern appears when you have to finish a task before you can continue with the next one sequentially.

Fig. 2 shows an example of the sequence pattern, Task A must finish before continuing Task B.

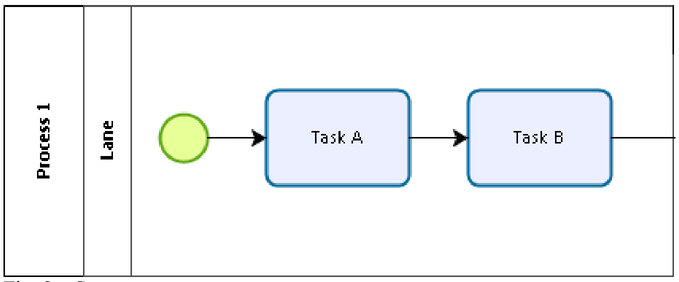


Fig. 2. Sequence pattern.

Exclusive decision pattern, this pattern appears when you must choose a single path among several ones that are available depending on a decision or process data. The gateway is represented by a rhombus.

Fig. 3 shows an example of the exclusive decision pattern, after the gateway, Task B, Task C or Task D can be chosen to continue with the process.

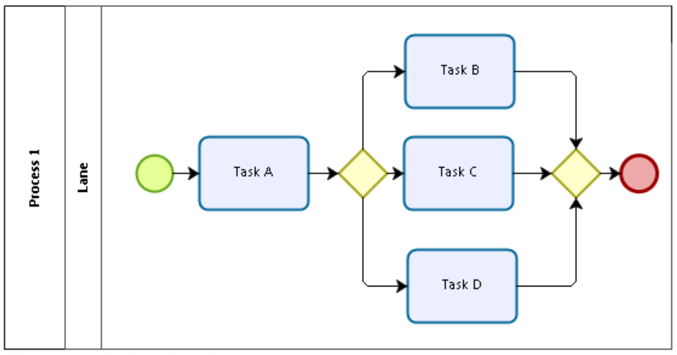


Fig. 3. Exclusive decision pattern.

Synchronization pattern, this pattern appears when two or more branches of the process are merged into one. It is expected that all incoming branches are completed before continuing the next task. The gateway is represented by a rhombus with a cross.

Fig. 4 shows an example of the synchronization pattern. Task B, Task C and Task D must be done after the gateway. You must complete the three tasks in order to continue with the process.

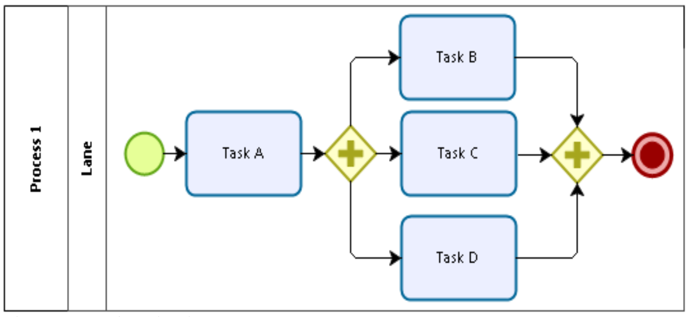


Fig. 4. Synchronization pattern.

Apart from these three patterns, it is also important to differentiate between the type of task, since each type has a different range of possibilities in the interface generation. This paper focuses on two types: user type tasks and service type tasks. **User type tasks** must be carried out by a person or user with the help of a system or software. **Service type tasks** are performed by a system without human intervention, for example an automatic task. [16]

According to the method of Fig. 1, once we have the patterns to analyze, next we study how these patterns are transformed into user interfaces. For this aim, we have used 7 projects of Bizagi [6], since it is a repository with BPMN models and interfaces that implement such models. Bizagi has also a suite with two complementary products, a Process Modeler and a BPM suite. Since 1989, Bizagi Process Modeler is a freeware software used to diagram, document and simulate processes using the standard BPMN notation:

In this article, 7 Bizagi projects were used to analyze how interfaces have been implemented from the three patterns. For each project, we have accessed to the BPMN models that represent the processes and to the final user interfaces that implements such processes. This way we can compare both models and interfaces to extract generation rules. Next, we describe each project briefly. This choice includes projects with several levels of complexity with application on different contexts in order to extract knowledge as general as possible:

- Project 1: Recruitment and Personnel Integration, the process reduces the processing time in activities such as informing the new employee about the documentation required for signing the contract, request of access, creation of users, creation of activities for the first week of the employee, among other things. The process helps the integration of new employees within the company, preparing them to perform their duties appropriately from the first day.
- Project 2: Change Management, the process responds to the need of introducing changes as part of the strategy of business continuity, new requirements and continuous improvement, determining and minimizing its impact through proper planning and control of its execution. The information that must be registered in each activity and the traceability of each change will allow you to prioritize and respond efficiently to requests. Additionally, you can reduce unsuccessful changes, satisfy external requirements, improve estimates of time and cost of projects, contribute to productivity, reduce response times and capture valuable information for continuous improvement.
- Project 3: Automobile Policy Subscription, the process begins with the entry of basic information of the policyholder, the insured, and the vehicle; going through the collection of complementary information, driver's record, claims of recent years, data of the last policy or the current one; the selection of a product for the policy or the type of coverage that you want to have, the selection of vehicle protections and your tax-deductibles, as well as the quote of the policy, the verification of risks, the control of delivery of documents, the verification and analysis of the final result of the inspection, and finally the issuance of the insurance policy.
- Project 4: Vacation Application, the process starts when any employee of the organization submits a vacation application, once it is registered, it is received by the immediate supervisor of the employee requesting the vacation. The supervisor must approve or reject the request; if the request is rejected, the application sends a message with the rejection information. Otherwise, if the request is approved, it will go to the final review by the Human Resources Department so that the update can be executed later in the system.
- Project 5: Invoice Payment, the process allows to receive invoices with all the information that is entered by reception, the financial assistant validates the invoice to perform the correct calculations, verifies if the invoice agrees with the information of the Purchase Order, and updates the financial ERP.
- Project 6: Disenrollment of personnel, the process helps the Human Resources area in the execution of the tasks related to the process. It begins when a Chief receives a resignation letter or decides to dismiss any of his employees. The process continues until the employee receives the settlement. Once the cancellation or deactivation of access permissions to folders and servers is done, the disaffiliation of the employee must be managed to the entities providing health, pension and other services related to social security. The employee's liquidation is performed according to the given conditions.
- Project 7: Contract Management, the process begins when a contract is required and its owner creates an application that will be evaluated by internal employees and previously defined participants (who act as representatives of the external parts of the contract) and who, as a result, will obtain a contract in execution. Any modification of the status of the contract, alarms of scheduled meetings and follow-up are supported. However, this process does not support the direct management of the document itself, its name, terms, conditions, policies and restrictions.

The most extensive and complex projects are Project 2, Project 3, Project 5 and Project 7 and the simplest ones are Project 1, Project 4 and Project 6. The level of complexity depends on the number of conceptual primitives in the business process model.

IV. Identification of Rules to Generate Interfaces from BPMN Patterns

Each one of the 7 Projects was analyzed to identify transformation rules from BPMN models to interfaces. These rules were extracted from the use of the 3 BPMN patterns within the Bizagi projects. We have tried to differentiate the different configuration options for each pattern that will generate different visual features in the interfaces.

The BPMN model has information about the behavior of the system but lacks of important information for the interface generation: the data. So, the interface generation cannot be tackled only with a BPMN model but it must be complemented with a Class Diagram in UML notation [17]. From the BPMN model we can extract information about the navigation among interfaces and their behavior, while the Class Diagram can be used to identify which information is required in each interface. That is the reason why we use also elements of the Class Diagram in our proposal.

Next we describe the rules to generate interfaces from models that were identified in the 7 Bizagi project.

Rule 0 and Rule 1 apply in all the 3 BPMN patterns.

R0: We have identified that in all the projects, a lane usually represents a form. From the attributes of a class diagram we can extract the fields of the form. In the form, there are input fields of different types: (1) input fields that accept any type of textual string (Textbox) [18]; (2) input fields that are a closed list where the user must select one element (Combobox or Listbox) [19], (3) input fields that represent a Boolean value (Radiobutton or Checkbox) [20].

R1: We have identified that in all the projects, user type tasks usually are transformed into a form. The input fields of the form are extracted from the attributes of the classes, each attribute generates an input field. R1 is complemented with R0.

Sequence pattern:

R2: We have identified that in Project 1, Project 2 and Project 3, if user type tasks are in the same lane and there is a dependency among them, this lane generates interfaces to guide the user through a process of several steps. This can be shown in different ways: (1) Wizard: the navigation throughout the different forms is forced to be done in a specific order. The number of forms depends on the number of user type tasks and each form is a user type task in sequential order [21]; (2) TabControl: a form with multiple tabs such a way each user type task generates a tab [22]; (3) Groupbox: the number of groupbox depends on the number of user type tasks. Groupbox can be generated such a way each user type task in sequential order generates a groupbox. [23] For the options (2) and (3) a form containing a tabcontrol or a groupbox must be created. For all options mentioned, the input fields of the form are extracted from the attributes of the classes, each attribute generates an input field. This rule is complemented with R0.

R3: We have identified that in Project 1 and Project 2, if user type tasks are in different lanes and there is a dependency among them, each lane generates a form. The creation of the form is done according to the order of sequence of each task; the input fields of the form are extracted from the attributes of the classes, each attribute generates an input field. This rule is complemented with R0.

R4: We have identified that in Project 3 and Project 6, if task A is user type and task B is service type and there is a dependency between them, then task A leads to generate a form. The data registered in the form of task A can be processed in task B by a web service or an automatic task. Both web services and automatic task can be shown in different ways in a form of two types: (1) report: designer of reports and graphics [24], (2) datagrid: it is a control that displays data in a customizable grid [25].

As summary, the extracted rules from the sequence pattern result in the generation of several user interface controls such as wizards, tabcontrols, groupboxes, reports, datagrids and forms.

Exclusive decision pattern:

R5: We have identified that in Project 1, Project 2, Project 3, Project 4, Project 5 and Project 7, the text that appears in the exclusive decision gateway usually represents a Label that includes a question such a way the user can take a decision. This control is associated with a previous user type task and components of Rule 6.

R6: We have identified that in Project 1, Project 2, Project 3, Project 4, Project 5 and Project 7, the text that appears in the connection objects of the exclusive decision gateway, generates a Radiobutton. Each option of the Radiobutton is a text of each connecting object of the gateway. This rule is associated with Rule 5. In the case that the gateway is manual, the rule does not apply.

R7: We have identified in Project 1 and Project 2 that if the task that continues in the exclusive decision gateway is of user type, it generates a form. The input fields of the form are extracted from the attributes of the classes, each attribute generates an input field. This rule is complemented with R0.

R8: We have identified that in Project 3, if a service type task A continues after the exclusive decision gateway, the previously registered data is processed in task A by a web service or a scheduled event. This rule uses the same options as Rule 4.

As summary, the rules extracted from the exclusive decision pattern result in the generation of user interface controls such as forms, labels, radiobuttons, reports and datagrids.

Synchronization pattern:

R9: We have identified that in Project 1 and Project 6, if after the synchronization gateway there are user type tasks that are in the same lane and there is a dependency among them, interfaces to guide the user through a process of several steps are generated. The alternatives to represent the different steps are exactly the same as the alternatives explained in Rule 2: Wizard, TabControl and GroupBox.

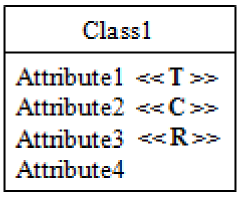
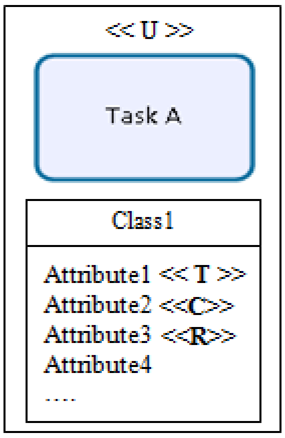
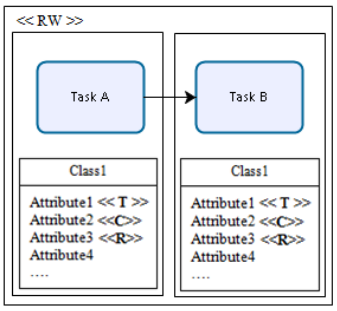
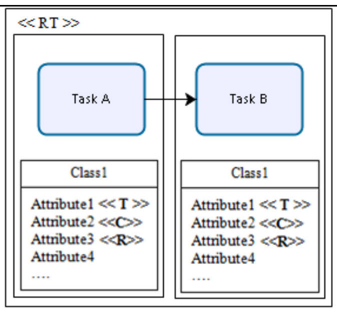
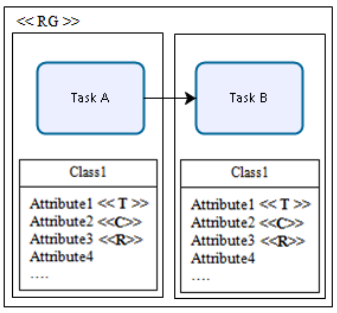
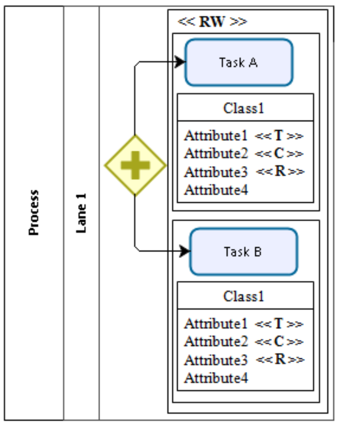
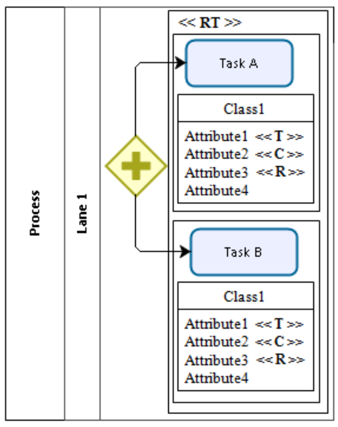
R10: We have identified that in Project 1, Project 6 and Project 7, if after the synchronization gateway the user tasks are in different lanes, each lane generates a form. The input fields of the form are extracted from the attributes of the classes, each attribute generates an input field. This rule is complemented with R0.

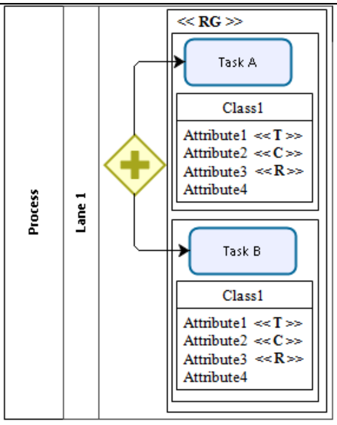
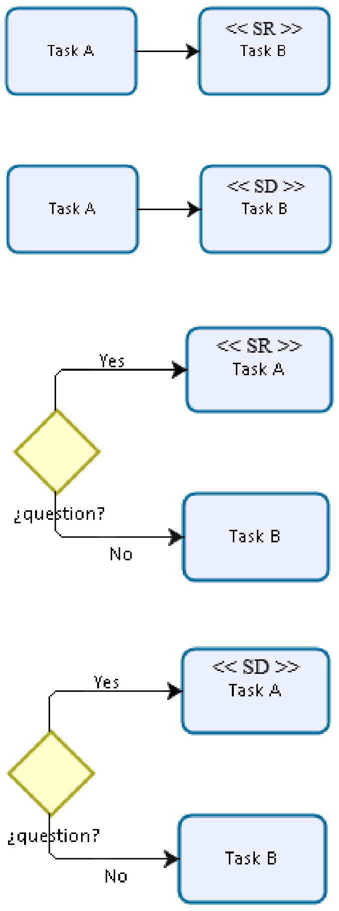
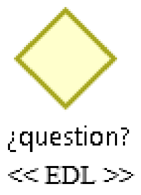
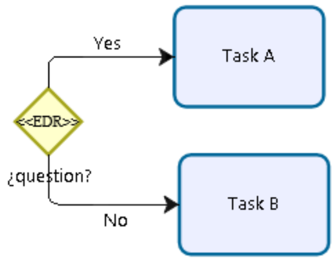
As summary, the rules extracted from the synchronization pattern result in generating user interface controls such as wizards, tabcontrols, groupboxes and forms.

V. Rules and Stereotypes to BPMN

We have seen in the previous section that the same pattern can generate different graphical representations. For each possibility of graphical representation we are going to define a stereotype so that the generation rules can be applied in an unambiguous way. TABLE I shows a summary of rules, stereotypes and generated user interface controls generated from those stereotypes. This table aims to summarize how to generate interfaces using stereotypes in the transformation rules previously defined. Next we explain each stereotype.

TABLE I. Rules, Stereotypes and User interfaces controls

Rules	Stereotypes	User interfaces controls
R0	 Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4	<< T >> Textbox << C >> Combobox, Listbox << R >> Radiobutton, Checkbox
R1, R3, R7, R10	 << U >> Task A Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4	<< U >> Form
R2, R9	 << RW >> Task A Task B Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4 .... Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4	
	 << RT >> Task A Task B Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4 .... Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4 ....  << RG >> Task A Task B Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4 .... Class1 Attribute1 << T >> Attribute2 <<C>> Attribute3 <<R>> Attribute4 ....  Process Lane 1 << RW >> Task A Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4 Task B Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4  Process Lane 1 << RT >> Task A Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4 Task B Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4	<< RW >> Wizard << RT >> Tabcontrol << RG >> Groupbox

	 Process Lane 1 << RG >> Task A Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4 Task B Class1 Attribute1 << T >> Attribute2 << C >> Attribute3 << R >> Attribute4	
R4, R8	 Task A << SR >> Task B Task A << SD >> Task B Yes << SR >> Task A ¿question? No Task B Yes << SD >> Task A ¿question? No Task B	<< SR >> Report << SD >> Datagrid
R5		<< EDL >> Label
R6		<< EDR >> Radiobutton

The classes that appear are stereotypes that represent a subset of the class model that is used in the task. The stereotypes << T >>, << C >> and << R >> can be used in the attributes of the class stereotype, << T >> represents the generation of Textbox, << C >> represents the generation of Combobox or Listbox, << R >> represents the generation of Radiobutton or Checkbox. These class stereotypes are associated with the user type task and can be used for Rule 0.

The stereotypes << U >>, << RW >>, << RT >> and << RG >> are represented through a table that contains a user type task and classes. These stereotypes are complemented with the attributes of the class diagram using Rule 0. << U >> represents the generation of a form, this stereotype can be used for Rules 1, 3, 7 and 10. << RW >> represents the generation of a wizard. << RT >> represents the generation of a tabcontrol. << RG >> represents generation of a groupbox. These stereotypes can be used for Rules 2 and 9.

<< SR >> represent the generation of a report. << SD >> represent the generation of a datagrid. Both reports and datagrids are created within a new form. These stereotypes can be used for Rules 4 and 8.

<< EDL >> can be used in the gateway that belongs to the exclusive decision pattern. It represents the question to ask the user to know the next step in alternative navigations.

<< EDR >> can be used for the options of the Gateway primitive that belongs to the exclusive decision pattern, it can generate radiobuttons.

VI. Illustrative Example

To illustrate the use of the transformation rules and the stereotypes, we will use Project 5 Invoice Payment from Bizagi. This project helps improve the reception, verification and approval of an invoice, reducing processing times and avoiding erroneous information. First we show the model as it was defined originally in Bizagi and next we enhance the model with our proposed stereotypes to generate its interface.

Fig. 5 shows the original BPMN diagram that represents the process Invoice Payment. It includes 4 lanes: Reception, Financial Assistant, Administrative Manager and Accounting. Each lane contains tasks: Receive Invoice, Validate Invoice, Return Invoice to Supplier, Approve products / services and Update Financial ERP. There are also gateways: Gateway Invoice match with PO, Products / Services Require Approval and Products / Services have been approved.

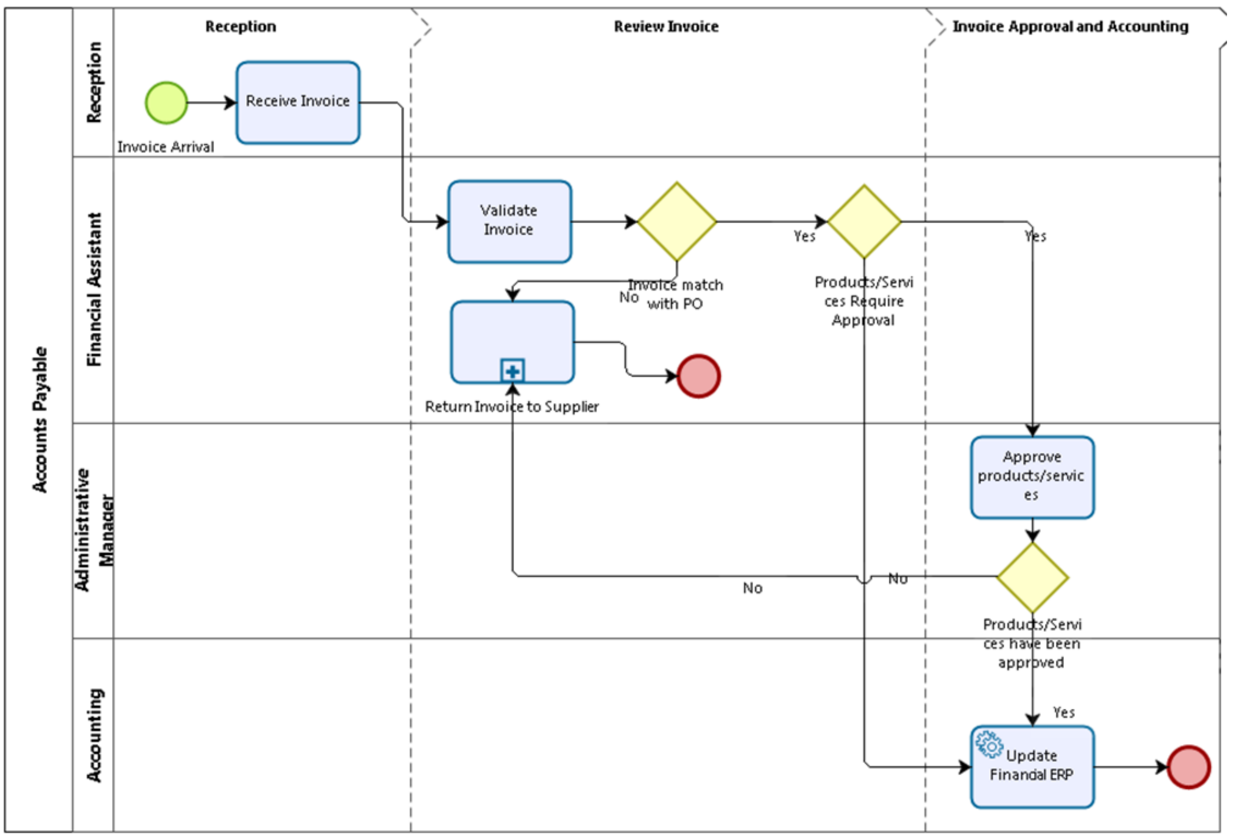


Fig. 5. BPMN model of Project 5.

In order to apply the rules to generate interfaces we must include stereotypes in the BPMN model. Next we deal with the application of stereotypes in the model of Fig. 5.

Fig. 6 shows the user type task Receive Invoice with the stereotype << U >> and the classes used in the task execution: Invoice, Supplier and Purchase Order. Moreover, each attribute of these classes has stereotypes to specify how the attribute will be displayed in the interface (for example <<T>>).

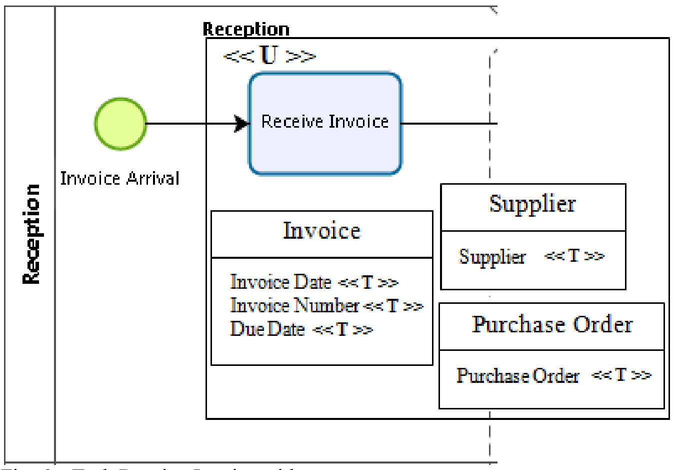
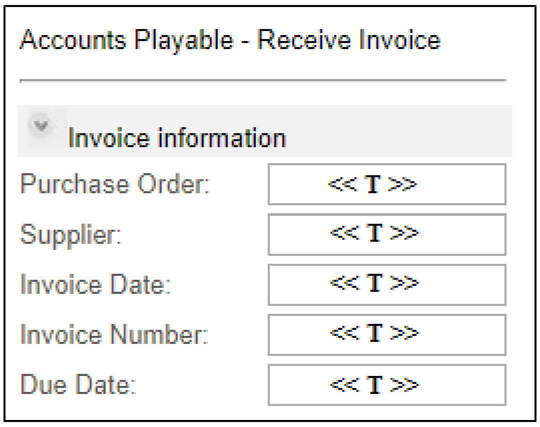


Fig. 6. Task Receive Invoice with stereotypes.

Fig. 7 shows the stereotype << T >>, which was applied in the user task Receive Invoice. It generates a form with the attributes of the class Purchase Order, Supplier, Invoice Date, Invoice Number and Due Date.


Accounts Playable - Receive Invoice

Invoice information

Purchase Order:	<< T >>
Supplier:	<< T >>
Invoice Date:	<< T >>
Invoice Number:	<< T >>
Due Date:	<< T >>

Fig. 7. Form Receive Invoice.

Fig. 8 shows the stereotype << U >>, which is applied in the user type task Validate Invoice. The attributes with stereotype << T >> are Invoice Number, Invoice Date, Due Date, Discount Date, Discount Percentage and Discount Date in class Invoice; Supplier, IdNumber, PaymentOption, PaymentTerm in class Supplier; Description, Quantitym UnitPrice and TotalPrice in class Invoice Product. All these attributes have the stereotype <<T>> since all of them are open fields to enter any string. The attribute Discount in class

Invoice has the stereotype << R >> since it is a boolean field. Cost Center has the stereotype << C >> since it is a closed list.

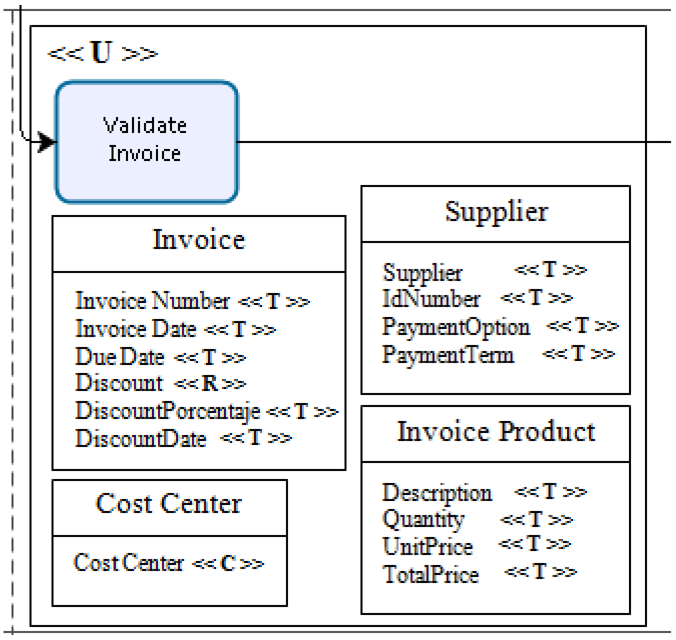
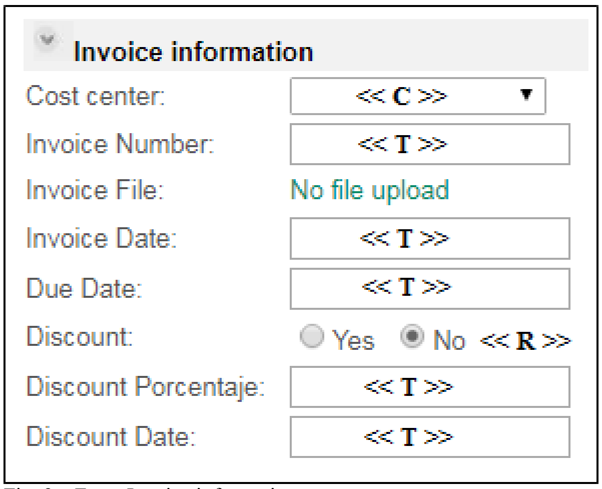


Fig. 8. Task Validate Invoice.

Fig. 9 shows the Invoice information form that belongs to the task Validate Invoice. This task generates a form to enter information about invoice creation. The form was created with the attributes of the class diagram; the controls satisfy the stereotypes used in the Invoice class. << T >> generates Textboxes, << C >> generates Comboboxes and << R >> generates Radiobuttons. The labels are the names of the class attributes: Cost center, Invoice number, Invoice Date, Due Date, Discount, Discount Percentage and Discount Date



Invoice information

Cost center:	<< C >> ▾
Invoice Number:	<< T >>
Invoice File:	No file upload
Invoice Date:	<< T >>
Due Date:	<< T >>
Discount:	Yes [✓ No] << R >>
Discount Porcentaje:	<< T >>
Discount Date:	<< T >>

Fig. 9. Form Invoice information.

Fig. 10 shows the gateway Products / Service require Approval?, this gateway has the stereotype << EDL >> at the bottom and << EDR >> inside the gateway. This gateway corresponds to the exclusive decision pattern.

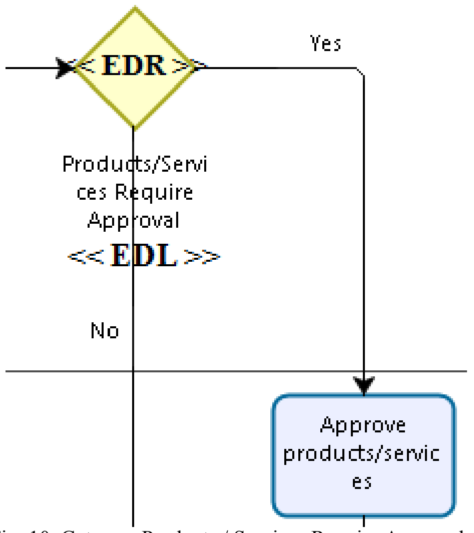
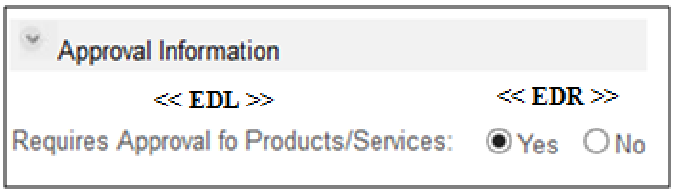


Fig. 10. Gateway Products / Services Require Approval.

Fig. 11 shows << EDL >> stereotype, which generates a label that is the gateway question. << EDR >> generates Radiobutton with two options Yes / No, which are the texts of the connection objects of the exclusive decision gateway.



Approval Information

<< EDL >>	<< EDR >>
Requires Approval fo Products/Services:	[✓ Yes] No

Fig. 11. Form Require Approval for Products/Services

As conclusion, we have a set of rules and stereotypes to generate interfaces. Note that the functionality behind these interfaces is not dealt with in our approach. Moreover, we have not yet implemented the transformation rules in a real model compiler to apply model to code generation. We have just analyzed existing projects to define transformation rules.

VII. Early Validation

This section describes an early validation that we conducted to check the applicability of our proposal. The validation consists in studying other 7 BPMN models different from the Bizagi repository to analyze the applicability of our rules. We have evaluated several elements of our proposal: (1) We have checked how much portion of the model is covered by the three patterns that we have used in our proposal; (2) We have also studied how many transformation rules can be used in those BPMN models; (3) We analyze possible discrepancies between the real interface that supports the implementation of the BPMN models and the interface that will be generated through our rules.

The BPMN models used as objects for the validation are 7 models that represent internal processes of the University Alas Peruvians. The BPMN models are Academic charge (AC), Registration enrollment (RE), Reservation enrollment (RSE), Update enrollment (UE), Validation of courses (VC), Change of modality (CM) and Certificate of studies (CE). Next, we start with the analysis of the use of the three BPMN patterns used in our proposal: Sequence Pattern, Exclusive Decision Pattern and Synchronization Pattern.

For each BPMN model, we have studied which percentage of tasks can be classified into one of the three patterns. TABLE II shows the following results, (1) Sequence Pattern is the most frequently used pattern. In the 7 analyzed models, the model with the highest percentage of use of this pattern is AC with a percentage of use of 75%; (2) Exclusive Decision Pattern is the second pattern that appears most frequently. The model with the highest percentage of use of this pattern is VC with a percentage of 43%. (3) Synchronization Pattern is the least used pattern. Both VC and CE are the two models with more use of this pattern (14%). Beyond the use of these three patterns used in our approach, we have identified other patterns that appear in the analyzed models. There are models like UE where the percentage of other patterns is 33%. In the column "Others" we identified 2 patterns based on events (implicit decision pattern and parallel execution pattern interleaved) and one basic flow control pattern (simple union pattern). These new patterns cover a small portion of the BPMN models but we should consider them in future works in order to support the generation of all the interfaces.

This shows that, even in the worst scenario, the percentage of use of the three patterns is higher to 65%. That means that only with the three patterns used in our approach we can generate most part of the interfaces.

TABLE II. PERCENTAGE PATTERNS IN THE MODELS

BPMN Model	**Sequence**	**Exclusive Decision**	**Synchronization**	**Other**
AC	75%	25%	0%	0%
RE	40%	30%	10%	20%
RSE	60%	20%	0%	20%
UE	50%	17%	0%	33%
VC	29%	43%	14%	14%
CM	67%	33%	0%	0%
CE	43%	29%	14%	14%

Next, we analyze how many rules of the proposed set could be used in the BPMN models of the 7 objects analyzed in the validation.

TABLE III shows the rules of our approach and the models analyzed. For each rule we have checked whether or not it could be applied to the model of the project. (1) We see that the rules that could be used in every BPMN model were R0, R1 and R3 because they are identified with the user type task that is very common in the processes; (2) rules R2 and R7 appears in most BPMN models. R2 is used so frequently since most models have user type tasks and dependencies in the same lane. Regarding R7, it is also so frequently used because there are many user type tasks after an exclusive decision gateway; (3) Rules R4, R5, R6 and R9 are used in half of the models. Rule R4 is used so occasionally due to in the 7 models we have found three service type tasks after a user type task with dependencies. R5 and R6 are used punctually since there are only four models that use exclusive decision gateways. R9 appears punctually since service type tasks after synchronization gateways is used only three times; (4) the rules R8 and R10 only appears in one model. These rules belong to Synchronization Pattern, which appears only in few portions of BPMN models.

Note importantly that all these rules appear in the models used in the validation because all these models use the three patterns that compose our approach. This also reinforces the idea that the three used patterns are widely used in any BPMN model.

TABLE III. RULES IN ACADEMICS PROCESSES

BPMN Models	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
AC	X	X	X	X	O	O	O	X	O	O	O
RE	X	X	X	X	X	X	X	O	O	X	O
RSE	X	X	X	X	O	X	X	X	O	O	O
UE	X	X	X	X	O	O	O	X	O	O	O
VC	X	X	X	X	X	X	X	X	X	X	O
CM	X	X	O	X	O	X	X	O	O	O	X
CE	X	X	O	X	X	O	O	X	O	X	O
Total	7	7	5	7	3	4	4	5	1	3	1

Finally, we check whether the real interfaces that implement the BPMN models agree with the interfaces that would be generated with our proposed rules.

Fig. 12 shows the percentage of agreement between the real interface and the interface generated with our rules. We can conclude that most user interfaces have an agreement higher than 70%. Since we identified other patterns that were not covered with our proposal, we cannot get values closer to 100% of agreement. The elements with the highest agreement between the real interfaces and the generated interface are forms, wizards, textboxes and comboboxes, since these elements are the most frequently used in the Bizagi projects used to extract the rules. We have identified several interface widgets that are not yet supported by our rules. Some of these elements are: (1) Datagrid, that is shown in several forms used for records; (2) Menustrip, that shows name of forms to access in the upper part of the main form [26]; (3) Timer, that uses time ranges to perform an action [27]. All these widgets could be generated with new rules to be identified in patterns different form the three patterns used in our approach. This work will be done as future research.

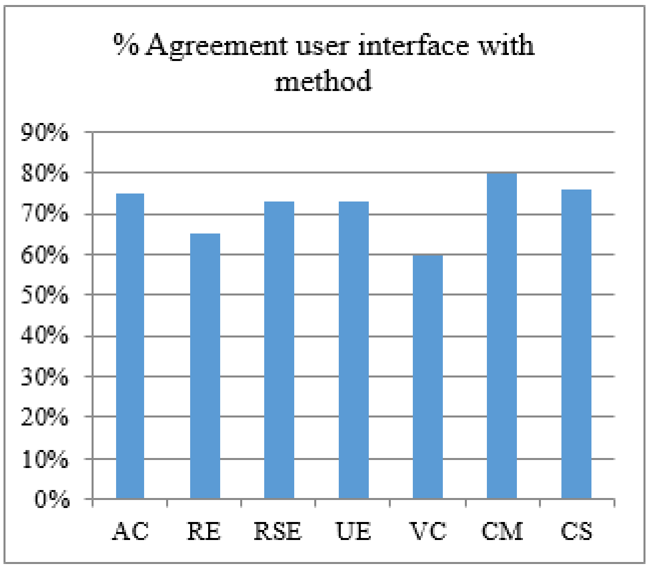


Fig. 12. Agreement user interface with method.

VIII. Conclusions

This article presents a method to generate user interface widgets from BPMN. First, we have analyzed 7 real projects to identify transformation rules from BPMN models to interfaces. These transformations have been focused on the use of three widely-used BPMN patterns: Sequence Pattern, Exclusive Decision Pattern and Synchronization Pattern. Second, we have defined a list of stereotypes for those generation rules with more than one alternative. Third, we have illustrated the applicability of the proposal with an example to show how transformation rules and stereotypes could generate interfaces. Fourth, we have conducted an early validation with other BPMN models different from the models used to extract the transformation rules.

The approach has some limitations. One limitation is that there is a strong dependency between BPMN and class diagrams. Not every BPMN model has a class diagrams to represent its persistency, reducing the applicability of our approach. Another limitation is that the early validation is only based on seven projects that are not much large (even though they are real). The results of the validation cannot be generalized to any model. Since the approach only focuses on the identification of transformation rules and the definition of stereotypes, we still could not include those transformations in a model compiler to validate the approach in a real generation of interfaces.

The goal of the approach in long term is to get a holistic software development method where we can generate as much as possible from BPMN models. However, note importantly that this work only focuses on the interface generation. How the interface triggers functions and how these functions are extracted from models is part of future work. Another future work is the implementation of these transformation rules in a model compiler. This way, we can generate real interfaces according to such rules. This will allow us to conduct a real validation of the approach in a real context.

Acknowledgements

This paper has the support of Generalitat Valenciana through project IDEO (PROMETEOII/2014/039), Spanish Ministry of Science and Innovation through project DataME (ref: TIN2016-80811-P) and Ministry of Education of Peru with the National Scholarship and Educational Loan Program PRONABEC - President of the Republic Scholarship.

References

[1] BPMN. *Business Process Modeling Notation*. Available: http://www.bpmn.org

[2] T. Allweyer, *BPMN 2.0: Introduction to the Standard for Business Process Modeling*, 2nd ed., 2016.

[3] Bizagi. (2002). *Bizagi*. Available: https://www.bizagi.com/es

[4] B. Group. (2017). *Adonis NP*. Available: https://es.boc-group.com/adonis-E

[5] IBM. (2017). *Auraportal*. Available: https://www.auraportal.com/es/

[6] Bizagi. (2017). *Examples of BPMN Projects*. Available: https://www.bizagi.com/es/comunidad/process-xchange

[7] L. Han, W. Zhao, and J. Yang, "An approach towards user interface derivation from business process model," in *Communications in Computer and Information Science* vol. 602, ed, 2016, pp. 19-28.

[8] K. Sousa, H. Mendonça, and J. Vanderdonckt, "User Interface Derivation from Business Processes: A Model-Driven Approach for Organizational Engineering," in *TAMODIA*, Toulouse (France), 2007, pp. 112-125.

[9] C. Ouyang, M. Dumas, W. M. P. V. D. Aalst, A. H. M. T. Hofstede, and J. Mendling, "From business process models to process-oriented software systems," *ACM Trans. Softw. Eng. Methodol.,* vol. 19, pp. 1-37, 2009.

[10] W. Bouchelligua, A. Mahfoudhi, N. Mezhoudi, O. Daassi, and M. Abed, "User interfaces modelling of workflow information systems," in *Lecture Notes in Business Information Processing* vol. 63 LNBIP, ed, 2010, pp. 143-163.

[11] J. Gonzalez-Huerta, A. Boubaker, and H. Mili, "A business process re-engineering approach to transform BPMN models to software artifacts," in *Lecture Notes in Business Information Processing* vol. 289, ed, 2017, pp. 170-184.

[12] M. A. Kabir, Z. Xing, P. Chandrasekaran, and S. W. Lin, "Process Patterns: Reusable Design Artifacts for Business Process Models," in *Proceedings - International Computer Software and Applications Conference*, 2017, pp. 714-721.

[13] A. Boubaker, A. Leshob, H. Mili, and Y. Charif, *A pattern-based approach to extract REA value models from business process models*, 2017.

[14] A. Yousfi, R. Saidi, and A. K. Dey, "Variability patterns for business processes in BPMN,"

Information Systems and e-Business Management, vol. 14, pp. 443-467, August 01 2016.

[15] Bizagi. (2014). *Process modelling patterns*. Available: http://resources.bizagi.com/docs/Workflow_Patterns_using_BizAgi_Process_Modeler_Esp.pdf

[16] S. A., *BPMN Modeling and reference guide*, First Edition ed., 2008.

[17] UML. (2017). *Unified Modeling Language*. Available: http://www.uml.org/

[18] G. Perry, *Aprendiendo Visual Basic 6*, 1 ed., 1999.

[19] J. F. Ramírez, *Aprenda Visual Basic practicando*, Primera Edición ed., 2001.

[20] D. Chapman, *Visual C++ .NET*, First Edition ed., 2002.

[21] M. MacDonald, *Pro .Net 2.0 Windows Forms and Custom Control in VB 2005*, First Edition ed., 2006.

[22] E. Rubin, *Microsoft Net Compack Framework Kick Start*, First Edition ed., 2004.

[23] G. Bronson, *Introduction of Programming with Visual Basic Net*, First Edition ed., 2005.

[24] B. Bischof, *Crystal Reports Net Programming*, First Edition ed., 2004.

[25] M. Schmidt, *Microsoft Visual C# .Net 2003 Developers CookBook*, First Edition ed., 2004.

[26] A. Road, *.Net Programming Covering C# 2005, Visual Basic 2005, ASP.NET and .Net Framework* First Edition ed., 2008.

[27] M. D. Network. (2016). *Class Timer Windows forms*. Available: https://msdn.microsoft.com/es-es/library/system.windows.forms.timer(v=vs.110).aspx