BPLOM: BPM Level-Oriented Methodology for Incremental Business Process Modeling and Code Generation on Mobile Platforms

Jaime Solís-Martínez, Natalia García-Menéndez, B. Cristina Pelayo G-Bustelo and Juan Manuel Cueva Lovelle
Department of Computer Science, University of Oviedo, Spain

***Abstract* — The requirements engineering phase is the departure point for the development process of any kind of computer application, it determines the functionality needed in the working scenario of the program. Although this is a crucial point in application development, as incorrect requirement definition leads to costly error appearance in later stages of the development process, application domain experts' implication remains minor. In order to correct this scenario, business process modeling notations were introduced to favor business expert implication in this phase, but notation complexity prevents this participation to reach its ideal state. Hence, we promote the definition of a level oriented business process methodology, which encourages the adaptation of the modeling notation to the modeling and technical knowledge shown by the expert. This approach reduces the complexity found by domain experts and enables them to model their processes completely with a level of technical detail directly proportional to their knowledge.**

***Keywords* — Business Process Modeling, BPMN, process transformation, code generation.**

I. Introduction

Software development has seen some great changes in some of its methodologies and techniques, but some of its problems have remained unchanged since their appearance in the mid and late twentieth century. One of these lasting problems is related with requirements engineering, aspect of the software development process that has seen very little evolution. Sommerville and Kotonya stated in their study [1] that there are several problems related with this initial activity of the software development process. The first problem mentioned is that the requirements engineer is not an expert in the application domain being addressed. Another of the difficulties present in this phase of the software development cycle which is also mentioned by these two authors is the fact that natural language is ambiguous, which has also been confirmed by the work of Laue and Gadatsch [11].

This work has been partially funded by the Ministry of Industry, Energy and Tourism of Spain, by the Avanza2 plan and by the European Regional Development Fund through the project "GADE4ALL: Plataforma genérica para facilitar el desarrollo de videojuegos y software de entretenimiento multiplataforma". The code for the project is MITC-11-TSI-090302-2011-11.

The Business Process Management Initiative (BPMI) [24], a group inside the OMG [23], proposed a solution for this problematic situation: Business Process Modeling (BPM), a discipline promoting the implication of the domain experts in the requirements engineering process through the use of a modeling notation that lies between the domain experts' language and the computer experts' knowledge. In its proposal, the BPMI introduced Business Process Modeling Notation (BPMN) [25] as the standard notation for BPM. Although their intention was different, this standardization made the notation grow in size and complexity to the point where non-technical domain experts must undergo a training process in order to understand it and use it properly [5].

Due to this circumstance, some simplifications of BPMN have been conceived, with different success degrees; one of these simplifications is Simple BPMN (SBPMN) [6], defined during previous work at the University of Oviedo with the following objectives: reduce the number of symbols needed to model a process and raise the user's level of abstraction. SBPMN was used to generate applications in a similar scenario to the one presented in this paper [20], although in that case several adaptions of the model were necessary for the code to be generated. The results obtained by SBPMN in the tests carried out were satisfactory but we consider the number of symbols it offers to be too large for non-technical domain experts. In order to solve this situation our first objective is the definition of a BPM level oriented methodology, a system that enables the adaptation of a graphical modeling notation to the skill level presented by the business expert.

In order for our approach to be used in real scenarios we need to achieve two other objectives: encourage business experts to model their processes and enable quick generation of the applications supporting the models created by the experts. We intend to complete these two goals with the definition of two separate but closely related tools: BPLevel Modeler and BPLevel Generator. BPLevel Modeler is a business process modeling tool which supports the level oriented methodology and promotes the involvement of business experts in the requirements engineering process through a simple and intuitive user interface. On the other hand, BPLevel Generator is capable of analyzing the models created with BPLevel Modeler and generate a specific and custom application for each model; in this case the tool is

DOI: 10.9781/ijimai.2013.222

aimed at the computer experts in charge of developing the applications in each scenario.

A summary of the structure of our proposal is shown in Fig. 1, where the whole modeling and code generation process can be overviewed; as it is seen, business experts will be able to create their models through the use of our modeling software and the models generated will be handed to the corresponding IT technicians who will generate the custom applications using our code generation tool. These computer experts will also be involved in the configuration of the XML file containing the graphical details of the custom application, which at this moment needs to be done manually.

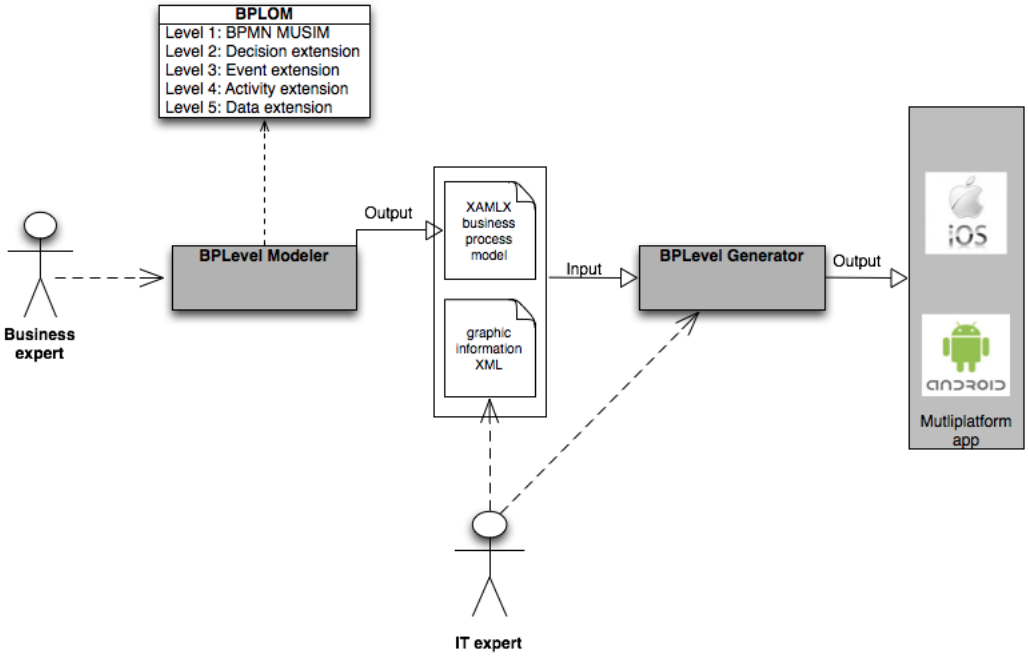

Fig. 1. Graphical overview of BPLOM use scenario

The rest of this paper will be structured as follows: section II will describe some of the existing BPM notations and establish the difficulties that non-technical domain experts undergo when using them. Section III will introduce our BPM level-oriented methodology and the results obtained by its initial level in a real scenario at a Spanish enterprise, where business experts used the initial level of our methodology for modeling their processes. In section IV we will present our tools, starting with BPLevel Modeler and continuing with BPLevel Generator and its intended use. Lastly, in section V, we will identify our conclusions and section VI will establish the future work we intend to carry out in order to improve not only our tools but also our methodology.

II. BPM and Business Process Modeling Notations

BPM is an initiative that encourages domain experts to define their business processes through modeling notations as a way of reducing the difficulties found in the requirements engineering process. This approach is based on the use of notations that are half way between the domain experts' language and the computer experts' knowledge.

There are, mainly, two types of business process modeling notations: graphical notations and textual notations. Bearing in mind our current application scenario, we have studied what we consider to be a representative set of notations that meet all of the following factors:

- High degree of diffusion, including in our revision those notations with higher diffusion degree.
- Wide range of complexity, from the most complex example (BPMN) to the simplest one (SBPMN).
- Domain expert implication, factor which rules out textual notations as a suitable modeling alternative. As stated by Lu and Sadiq [10], graphical notations allow users to represent business processes through simpler semantics and more abstract syntax, circumstance that lowers the complexity domain experts experience during the modeling and verification of the processes.

A. BPMN: the standard notation

BPMN is the standard notation for business process modeling. It was proposed by the BPMI in 2004 and since then it has undergone periodical revisions, being at the time in its 2.0 version. It is the most widely use notation of this type, with more than 70 implementations nowadays.

The two objectives that where stated as principal goals for BPMN in the introductory document written by Stephen White [2] where the following: provide domain experts an understandable and usable notation and reduce the number of existing notations and modeling tools.

1) BPMN's features

BPMN's root element is the Business Process Diagram (BPD), which is composed by a set of activities that represent the actions present in the business process and a group of flow

controls entities that establish the order in which these activities are done. The models built with BPMN can be enriched with other elements such as events, choreographies, messages, lanes and pools. BPMN's elements can be divided into the following categories: activities, gateways, conversations, choreographies, events, swimlanes and data.

The number of elements present in the BPMN specification is large, circumstance explained by the standard category of the notation. Version 1.1 of BPMN consisted of 52 different elements whilst the actual 2.0 version has seen an increase in this feature, as it can be seen in the BPMN 1.1 and 2.0 posters linked in the BPMN web page [25].

2) BPMN's disadvantages

The main issue regarding the use of BPMN when dealing with non-technical domain experts for process modeling is its complexity.

Wahl and Sindre have confirmed this fact in a study [5] where they state that a non-technical expert will need training in order to be able to use BPMN in a correct way. This complexity can also be seen in the investigation carried out by Recker in 2007 [7], where through the answers of 590 BPMN users he has been able to establish that there are several entities in BPMN that receive very little use. This fact can be seen in Fig. 2, a graphic classification of BPMN entities based on the use they receive obtained from the results presented in the referred work. The entities in BPMN are classified into three different categories: important, sometimes used and unused.

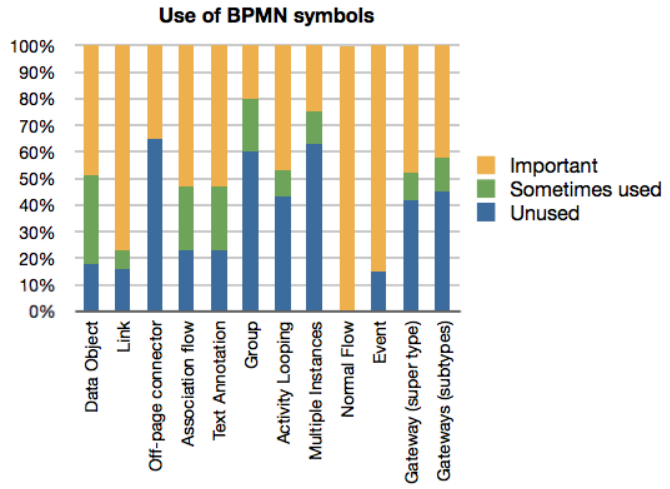

Fig. 2. Graphical representation of BPMN symbol importance obtained in Wahl's and Sindre's work

Stephen White and Miers also mention BPMN's complexity in their BPMN reference guide [8]. In one of the sections of the book they point out that it is not likely for a business analyst or end user, who can also be referred to as a application domain expert, to need all the symbols included in BPMN.

Another issue concerning the usage of BPMN by non-technical domain experts is the great number of BPMN supporting tools that exist actually. Although one of the objectives stated in the presentation of BPMN [2] was to reduce the number of tools with support for BPMN, the situation has gone the other way; this circumstance can be confirmed by a study carried out in 2010 [13] where it is stated that the popularity of BPMN has encouraged the appearance of a greater number of modeling tools with support for the standard. The magnitude of this problem seems to increase when the differences between the tools are notable due to the interpretation of the standard made by the authors and the different approaches that lead to including BPMN entities to the tool or not.

B. UML Activity Diagrams

UML Activity Diagrams, which can also be referred to as UML AD, are one of the types of diagram included in the Universal Modeling Language (UML) [26] specification.

The Activity entity, which represents the different actions that have to be carried out during the execution of the process, is the base of this type of diagrams. The activity entity is accompanied in these diagrams by other artifacts like decisions and parallel activity execution syntax. Although UML AD include entities present in the other business process modeling notations, a study [4] referenced in our analysis establishes that UML AD are less expressive than BPMN.

Even though UML AD do not include such a great amount of entities as BPMN, fact which enables a reduction of the complexity found when modeling business processes, there are some issues regarding the graphical representation chosen for them. This circumstance is triggered by the fact that some of the entities included in the specification share the same graphical representation. For example, the decision entity is represented in the same way as the merge entity, situation that can lead to problems for the understanding of the model by the non-technical domain experts. This circumstance can be seen in Fig. 3, which shows a sample UML Activity Diagram process.

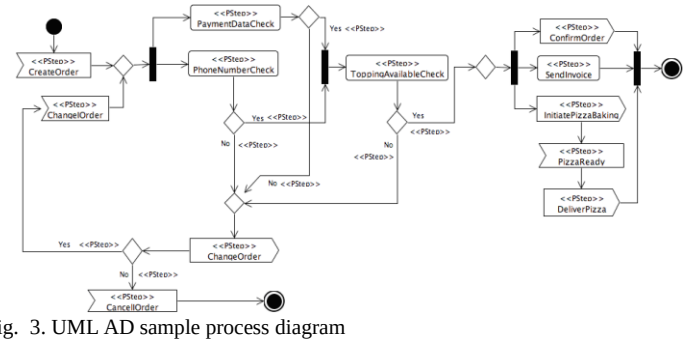

Fig. 3. UML AD sample process diagram

The other aspect that encourages us to avoid using UML AD as a valid notation for process modeling by non-technical domain experts is its abstraction level. As UML is a general-purpose language, UML AD offers the user a very low abstraction level. This aspect goes against one of the main goals in this investigation: usability experienced by non-technical domain experts.

C. jPDL, process modeling language in Java

jPDL, which stands for Java Process Definition Language, is a graphical language for business process definition included

in jBoss jBPM [27], a BPM suite written in Java for applying BPM under this platform.

This language appeared as a simplification of BPMN via two different approaches: reducing the number of entities available for the definition of the process and modifying some of the graphical representation of the entities through a color code. As a result of the appliance of these approaches, jPDL has managed to offer a lower complexity level than BPMN and has also achieved an increase in the user's level of abstraction. Fig. 4 shows a sample process diagram built using jPDL, where the color code can be appreciated as the main difference between this diagram and the one generated with UML AD.

An example where jBPM is used to increase the level of automation of the business processes can be found in the work done by Castaño [21]. Having correctly identified the suite's capability for adapting to nearly all business process, Castaño introduces a prototype using jBPM that enables further automation of business processes through the use of data mining. Via this solution, the dependency that some processes may have with human interaction can be reduced.

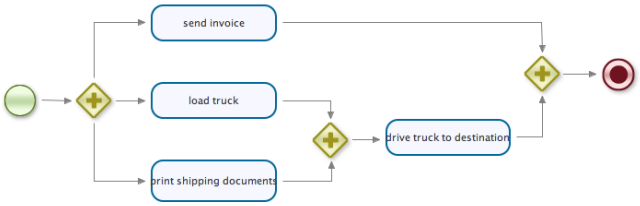

Fig. 4 jPDL sample process diagram

Although the inclusion of this color code favors the comprehension of the models built with jPDL and despite the fact that this language has managed to reduce the number of entities offered by BPMN, we consider that jPDL is not suitable for our investigation. Our main concern regarding jPDL is the user's level of abstraction, as there are two aspects related with the activity entity that make it low: on one hand, the fact that there are various types of activities which have high technical content (the script task, for example) and, on the other hand, the need to edit under some circumstances the XML code behind the graphical representation of the business process in order to configure some of its details.

D.Petri nets, another option for process modeling

Petri Nets [14], defined by Carl Adam Petri in the mid twentieth century, are another option to be considered when dealing with process modeling. There are several features that make Petri Nets suitable for this task, as stated by van der Aalst in his study [15]:

- Formal semantics, which enable the precise and clear definition of process models.
- Graphical nature, which promotes process definition through the use of nodes and transitions.
- Expressiveness, as they support all the primitives used when defining processes.
- The existence of many analysis techniques that can be applied to them. These techniques can be used to evaluate properties and also to calculate performance rates.
- Platform independency, as Petri Nets are not based on any proprietary software.

Despite these properties Petri Nets have and although they can be defined graphically, there is great concern when introducing them to non-technical domain experts: their complexity and low level of abstraction. These two inconveniences are against our usability and business expert focuses so we have decided to avoid using Petri Nets in our scenario, although we think they are suitable for completing modeling tasks under other use circumstances.

E. SBPMN: Simple BPMN

Simple BPMN, also referenced to as SBPMN [6][20], is a reduction of BPMN attempted as a previous investigation of components in our group at the University of Oviedo [28].

SBPMN arose as a possible solution to the problems found with the abstraction and complexity levels of the notations previously presented in this paper. One of the objectives of SBPMN was to reduce the technical knowledge level needed to complete the modeling of a business process by the domain expert. It also tries to avoid the arbitrary use domain experts give to some of the symbols present in BPMN.

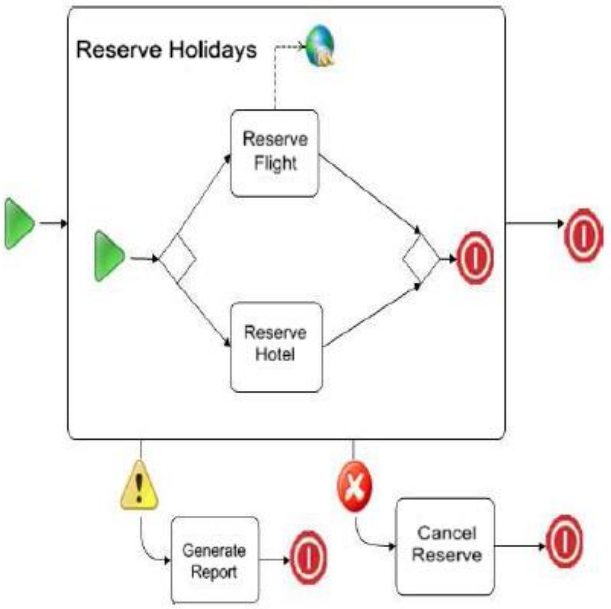

Fig. 5. SBPMN sample process diagram

Fig. 5 shows a sample SBPMN diagram representing the process used to make a trip reservation. As it can be seen, SBPMN offers different graphical representation of some of the entities it borrows from BPMN, which makes the process more intuitive for the user and simpler to understand.

The graph presented in Fig. 6 shows a graphical summary of the results obtained by SBPMN in the tests that were carried out after its definition. These results show that percentage of errors or failures made by domain experts when using SBPMN was less than when using BPMN, showing a reduction rate greater than 20%. This error reduction implies that the skill level shown by the users raised considerably when using SBPMN (more than 80% of skills shown) compared to BPMN (less than 50% of skills shown).

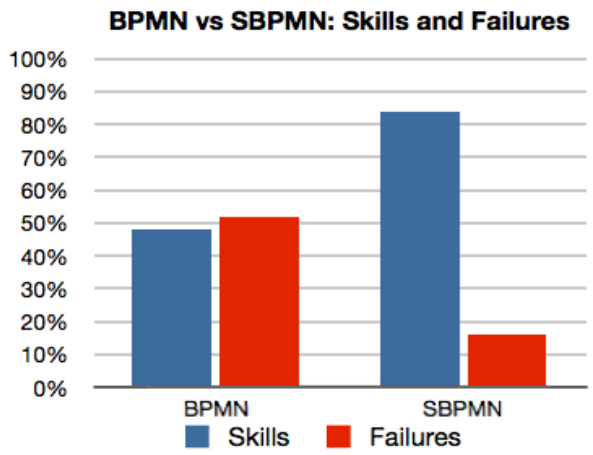

Fig. 6. Comparison of skills and failures between BPMN and SBPMN

The results shown establish that SBPMN is simpler than BPMN for non-technical domain experts when modeling their business processes. The problem concerning this simplification, despite the great effort shown in managing to simplify process modeling to non-technical domain experts, is the fact that the number of entities offered is still big and sometimes can be too complex for non-technical experts; for example, SBPMN proposes the use of various types of activities (human task, simple task and automatic task) and includes some entities with technical background like XML schema data object and data source object.

III. BPLOM: BPM Level Oriented Methodology

Until this moment, we have focused on analyzing several notations based on their degree of diffusion and their complexity. All of the notations presented have managed to model processes correctly but there are several issues which make us discard them as suitable for non-technical domain experts: low abstraction level, big complexity in their use and difficulty in understanding the models at first sight.

A common feature of all of these notations is to present the user a set of entities to use, which cover the basic and the advanced features that can be present in a process model. This circumstance can bring problems to novice non-technical users, as they face the use of complicated and technical artifacts (like events, signals and data objects) that they may not understand fully. This situation can also lead to misuse of some of the entities in the language and, thus, to the specification of a wrong model.

In order to prevent this situation, why not present the user a set of entities that is capable of adapting to his modeling skills and knowledge? This is the basic approach of our proposal: a level-oriented methodology for the application of BPM to any type of business process, promoting the adaptation of the modeling entity set to the expert's knowledge level. This departure point differs from other business process modeling notations like the one introduced by Chinosi and Trombetta in the 2009 edition of the BPM Handbook [18]. The first stage of the referenced methodology is based on reading the documentation related to the process and creating a primary sketched version of the process derived from the interpretation a computer expert has of this documentation. Although this approach could be valid, the ambiguous nature of natural language [1][11] does not recommend it.

Our proposed methodology will be divided into 5 incremental levels. Level 0, also known as BPMN MUSIM [16], will be the initial level and it will offer the minimum number of entities needed for business process modeling. Throughout the following levels we will be introducing more artifacts like events or data objects to the methodology in order for it to gain in expressiveness, trying to reach an expressive power as similar as possible to that in BPMN. The levels will be incremental, so the user will be able to model his process using the entities contained in the current level in addition to those contained in the previous ones.

This gain in expressiveness achieved when going up through the levels is directly related to an increase in the complexity users fin when using the symbols included in the methodology, but as the higher levels are intended for more technical based experts this complexity increase is manageable.

A. Features of the Methodology

The methodology we propose has some key features we would like to point out. These are the following:

- Incremental nature of the levels, which allow the user to model his processes with the entities that adapt to his modeling skills and technical knowledge.
- All processes can be transformed to code and executed, no matter in what level they are in.
- Platform independency. Despite the fact that we have chosen .NET as the target platform in our study, this approach could be used to generate code for any other platform.
- BPM phase schema reduced. As the domain experts are in charge of modeling the processes and these will be used to generate code, there is no need to capture requirements in text form and translate this text into models. This enables the avoidance of common errors in the requirements engineering process [1] and reduces the 5-phase BPM application scenario described by Ryan K. L. Ko [9].

B. Drawbacks of the Methodology

The main disadvantage of our methodology in comparison to BPMN is its expressive power. Although our methodology includes the majority of the artifacts and entities included in the standard, some other components of it have been dismissed due to different reasons (see discarded BPMN entities section of this paper).

Despite this fact, we think the methodology is still capable of modeling any type of business process completely and with no need for any additional symbols, notwithstanding the possibility of further extensions of the entity set available in our approach.

C. Level Description

Once our proposal's main features and limitations have been introduced, the next step is to present each of the five levels that make up our methodology. This presentation will be done by introducing the symbols or entities present in each of the

levels with an explanation of their functionality and an overview of their graphical representation.

1) Level 0: BPMN MUSIM

The first level of our methodology is called BPMN MUSIM [16], which stands for very simple BPMN. As the basis of our methodology, this level is intended for novice, non-technical domain experts that want to model their processes through a simple and clear notation. It contains the minimum set of symbols needed for process modeling, 5 entities in total. BPMN MUSIM's main feature as an introductory modeling artifact for novice, non-technical domain experts is the quick and simple learning process domain experts undergo before they start modeling with it.

a) BPMN MUSIM entity selection

As it was stated before, BPMN MUSIM is conceived as the minimum set of symbols needed to define a business process completely. The selection of the entities it includes has been made following the results of several studies [3][7][9]. Particularly interesting are the results of the study carried out by Recker in 2008 [3], which show, through the analysis of several BPMN models, that there is a common subset of entities that are present in the majority of BPMN models produced by experts.

Fig. 7 shows a graphical recreation of the results obtained by Recker in this study in the form of a set diagram. Each of the boxes holds a group of entities of the BPMN symbol set and a number indicating the amount of times these entities appear together in the studied BPMN process models. For example, 116 models have tasks and sequence flow entities in common and 65 processes have these two entities and the start and end events in common; the greater the number inside the box, the greater the chance a business process model has of containing all of the entities inside the box. As the graph shows, the most commonly used symbols in BPMN are: tasks, sequence flow, start event, end event, pools and gateways. Based on this study and bearing in mind pools are mainly used as the representation of the ownership a user has of the business process, we believe 5 entities can compose the minimum set allowing complete modeling of business processes and thus we propose it to be the introductory level for our methodology.

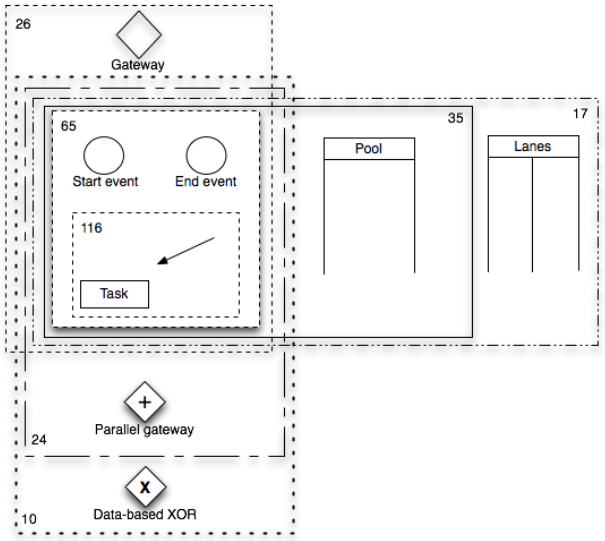

Fig. 7. Results extracted from Recker's 2008 study

b) Level 0 elements

The set of entities included in this level and their graphical representation is the following:

- **Starting point:** All processes modeled with BPMN MUSIM must have a single starting point that will be represented by a green circle.
- **Ending point:** A process defined with BPMN MUSIM can have one or more ending points, which represent the end of the process. The ending point in BPMN MUSIM will be a red circle.
- **Activity:** A rectangular shape with its name inside will represent an activity.
- **Transition:** A transition represents the flow between two elements of the model and will be represented with an arrow. The arrow's head will point the direction of the flow.
- **Decision:** Decisions enable alternative taking in business processes. A decision will be based in an expression to decide upon and two branches: true and false. A diamond shape will represent decisions.

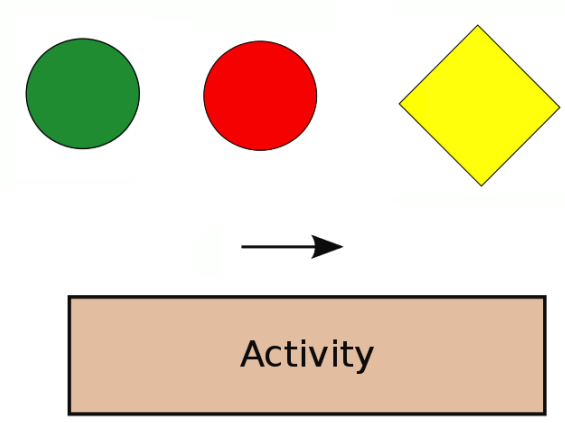

Fig. 8. Graphical representation of BPMN MUSIM entities

The graphical representation of the entities in this level, presented in Fig. X, was chosen due to the following reasons.

- The need to be close to the graphical representation chosen by the OMG for BPMN. As the standard notation, BPMN is widely used not only as it is but also as the basis for other notations and tools (like Microsoft Visio for example). For this reason, if we define a similar graphical representation for our initial level, user's migration to our approach can become a simpler task.
- We have defined first sight differentiable symbols for each of the entities offered in our methodology in order to avoid one of the problems found in UML AD, where two artifacts shared the same shape and this could lead to process experts misunderstanding the models.
- As the coloring approach offered by jPDL seemed to be generating more comprehensible models than those obtained with black and white entity representation, we decided to include color to our shapes. In this way, we continued with the green and red color code for representing starting and ending points and also added colors to the other symbols.
- Recker, Safrudin and Rosemann studied novice user modeling patterns in their work [12] and stated that this type of users understand better models including text and abstract symbols (circles, arrows and rectangles) than those containing concrete figures.

c) BPMN MUSIM's use in real life processes: examples and results

BPMN MUSIM has been used for modeling two real life processes in a Spanish enterprise, trying to demonstrate its suitability for business process modeling.

In order to establish its skills for the modeling of any kind of business process we decided to use BPMN MUSIM to model the following processes: informatics incidence management and recruitment, one of them close to computer science and the other concerning non-technical aspects. Details on the process models and other circumstances regarding the application of BPMN MUSIM to this use case can be found on the two articles [16][17] presented in 2011 at an Iberian congress.

The scenario designed for the application of BPMN MUSIM to these processes started with a brief meeting with the domain experts, where they were introduced to the notation: its entities, their meanings and the first modeling exercises. Once the experts understood the language and were capable of using it, an iterative meeting approach was taken: each of the experts was addressed to come to a meeting with the process models he had done and these would be reviewed with the project's team in order to spot the errors or difficulties; when the review was complete, the expert was set to correct the errors and another meeting was scheduled in the following days in order to undertake further review. This meeting schedule was repeated until the non-technical domain experts marked the process models as definitive.

As the following figures show, BPMN MUSIM has obtained good results in the aspects that were measured after its introduction to the domain experts. In first place, users have stated a better comprehension of the symbols in BPMN MUSIM, as seen in Fig. 9; this circumstance is due to the inclusion of the color code, which enables users to understand models better at first sight. Results also point out, like it is seen in Fig. 10, that the majority of the domain experts didn't spot the need for any additional symbols in BPMN MUSIM in order to be able to model their processes completely.

The interpretation of these results allows us to think that BPMN MUSIM symbols are easier to use than BPMN symbols and that the proposed symbol set is sufficient for non-technical experts to model their processes completely at a basic or early stage modeling level.

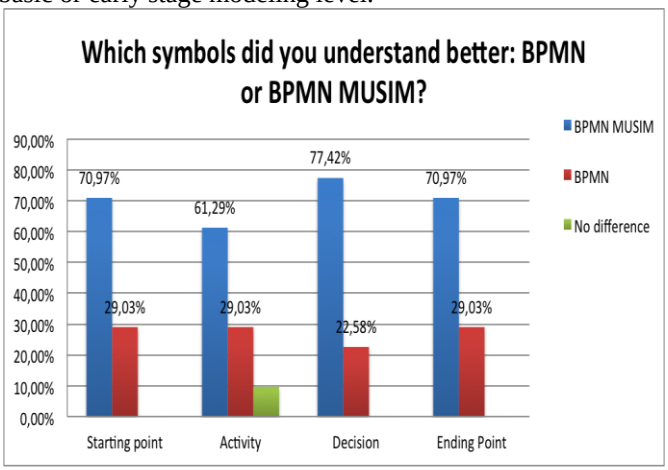

Fig. 9. BPMN MUSIM and BPMN symbol simplicity comparison

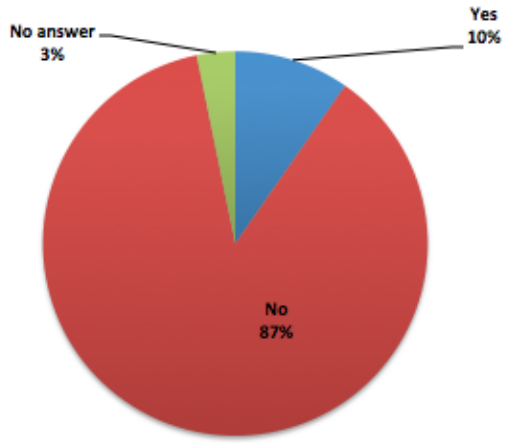

Fig. 10. Pie chart representing need for additional symbols in BPMN MUSIM

2) Level 1: Decision Extension

Once the basic symbols needed for process modeling have been introduced, its time for increasing the expressiveness of the methodology.

Fig. 7, which established the most widely used symbols in BPMN diagrams built by domain experts, showed that the main core of symbols are the ones included in BPMN MUSIM and that these are followed by the use of more complex decision entities. In order to follow the tendency pointed out in this investigation carried out by Muehlen and Recker [3], we have decided that the second level of our methodology is going to be the decision extension.

a) Level 1 elements

The symbol set proposed as a decision extension for the methodology is the following:

- **Parallel decision:** A parallel decision allows the expert to define the execution of two simultaneous paths inside the process. A yellow diamond shape with a cross inside will represent it.
- **Inclusive decision:** Inclusive decisions enable the activation of at least one of their branches, depending on the incoming condition. A yellow diamond with a circle inside will represent an inclusive decision.
- **Join:** The inclusion of these new decision types forces the inclusion of the join entity, the point where the process waits for the completion of the process' paths before moving forward to the next activity. A horizontal line with two incoming transitions and one outgoing transition will represent joins.

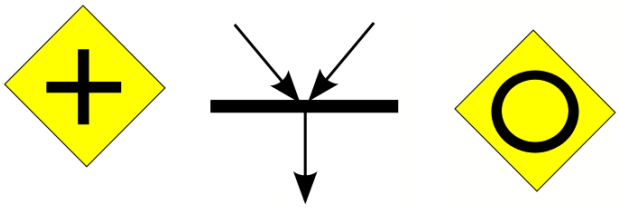

Fig. 11. Graphical representation of Level 1 entities

3) Level 2: Event Extension

Once the elements pointed out by Muehlen and Recker as the most used in BPMN have been included in the methodology, it is time to extend our proposal with other type of artifacts provided by BPMN and which are intended for more technical and experienced users.

BPMN gives a lot of importance to the events, entities that reflect the appearance or occurrence of certain actions that alter the normal process flow. The entities of this kind offered by BPMN cover from messages to signals, without forgetting others like time events and errors. With a closer look at the BPMN 2.0 entity set, clearly represented in the BPMN 2.0 poster offered by the OMG [23], there is a main issue regarding the event use in this version of the standard: each of the events proposed in BPMN 2.0 has several graphical representations depending on the place where they appear in the model (beginning, intermediate and end) and if they activate a subprocess or not.

In order to simplify the event model proposed by BPMN for the non-technical domain experts and, at the same time, avoid the appearance of different entities with similar graphical representation, we propose another approach for event modeling under level 2 of our methodology. We call this level the event extension.

a) Level 2 elements

The entities included in the event extension of the proposed business process modeling methodology are the following:

- **Message event:** This type of event allows the user to include message receiving and sending inside a process, enabling communication between different processes and/or users. Messages will have two graphical representations in this methodology: an outgoing message will be represented by an envelope with an arrow pointing up and incoming messages will be represented by an envelope and an arrow pointing down.
- **Time event:** A time event allows the definition of time conditions inside a process, like the fact that a process must wait for a certain activity to end before continuing or also a time lapse. A clock will represent time events.
- **Error event:** Error events define the place where a process stops due to the appearance of an error, ending the process' execution immediately. Error events will be represented by a prohibition signal with the word "Error" inside.
- **Cancelation event:** Cancelation events represent the moment where a process' execution is cancelled. A red colored cross will represent these events.
- **Signal event:** This event allows the user to send a signal to another process in order for it to continue its execution. It is directly related with the time events presented before. A danger signal will represent signal events in this methodology.

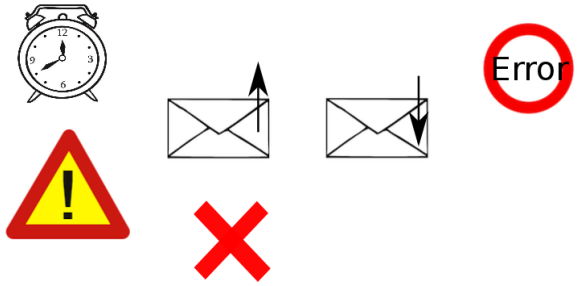

Fig. 12. Graphical representation of event extension entities

b) Using Events in the Models

Once the events have been introduced, there is the need to explain how they can be used for process modeling. All events, except cancelation and error events that need to appear just before an ending point, can appear in any point of the process.

Events are artifacts that make the process stop until their occurrence in order to continue normally. Thus, an event must always appear after the activity that generates it and before the entity whose execution it has to impact.

For example, if we need to model a process where an activity causes the sending of a message and after the delivery takes a decision we would have to place the activity first, then the outgoing message entity and finally the decision. Making the model like this will ensure the process will send the message once the preceding activity has been completed and also it will wait until the message is sent before going on to taking the decision.

4) Level 3: Activity Extension

The fourth level of the proposed methodology is defined as an activity extension for the proposal. Until this moment the methodology only provided the user one type of activity, which represented an automatic or user task.

However, as domain experts move along the levels of the methodology they have more experience with the use of the notation and also their processes need of more rich constructions in order to be modeled precisely. Under these circumstances, the activity extension of the methodology introduces subprocesses and function calling to its artifacts.

a) Level 3 elements

The elements included in the activity extension of the proposed methodology are the following:

- **Call activity:** A call activity references a global task that is repeated throughout several processes. This entity allows the user to call certain a certain task without having to redefine it in each of the processes it appears in; a common call activity could be user identification or login. Call activities are represented by the same entity as the normal activities but including a world globe icon that identifies it as a call activity.
- **Subprocess:** A subprocess is a set of activities that need to be done without any interruption. If any of the activities inside one subprocess produces an error, the process will be terminated. A subprocess is represented by a dashed rectangle that includes all of the activities of the subprocess.
- **Event subprocess:** Event subprocesses are a special type of subprocess, which is triggered after the appearance of an event; it will behave identically as the normal subprocess in case an error takes place. Its representation is the same as the normal subprocess one but the first entity of an event subprocess must be an event (in the following image the event is an incoming message event). The events available for an event subprocess are: incoming and outgoing message events, time events and signal events.

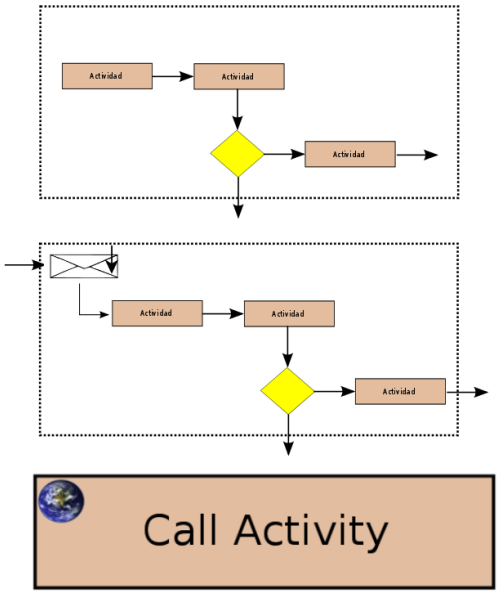

Fig. 13. Graphical representation of the activity extension elements

5) Level 4: Data Extension

The last level of this BPM methodology is defined as a data extension. Until this level, the introduction of technical detail in the models has been kept as small as possible in order to preserve the expert from tying the model to any technical aspect.

However, at this top level of the methodology the experts are considered both experienced modelers and technically prepared, so this status requires the introduction of data elements that enable the definition of information structure and flow through the processes. This detail is directly related with technical application implementation details that are too complex for the novice modelers using the lower levels of the methodology.

a) Level 4 elements

The elements included in the data extension level of the methodology are:

- **Data object:** A data object represents information that flows through the process in different ways (documents, data introduced in a form, etc.). A paper sheet that represents information stored in a computer defines a data object.
- **Object collection:** An object collection represents a set of data objects that flows through the process, like a list of documents for example. A stack of documents, which establish that an object collection is made up of multiple data objects, represents it.
- **Warehouse:** A warehouse represents the moment where a process reads or writes data in a database. This implies that data generated in a process and passed to a warehouse survives the process' instance. The classical hard drive representation with the word "Warehouse" inside will represent this entity.

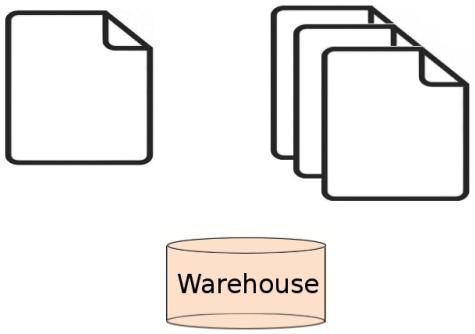

Fig. 14. Graphical representation of Level 4 entities

6) Discarded BPMN Entities

As it has been seen in the presentation of the levels that make up our methodology, there are several artifacts included in BPMN that are not present in our proposal. As the results of the referenced studies show [3][19] several of the modeling entities offered by the standard business process modeling notation experience little use.

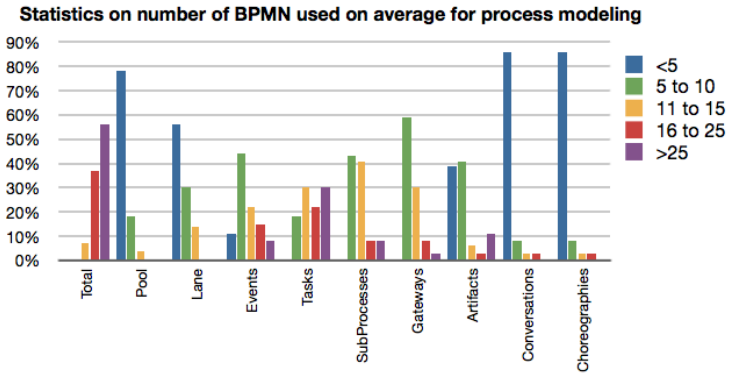

Fig. 15. Results obtained by Chinosi and Trombetta in their work

Fig. 15, a representation of the results from Chinosi's and Trombetta's study published in 2012 [19], shows a bar graph with a measurement for the number of times each of the represented BPMN constructs appear in the models included in the study. As it can be seen, BPMN elements like conversations, choreographies and pools, for example, experience low use by modeling experts. Thus, we have decided to exclude these symbols from this first version of our proposed methodology. Although the results extracted from this study differ in some of its figures from those obtained by Recker [3], the similar low symbol usage trends allow us to rule out some of the least used features found in BPMN.

At this moment we must establish that the exclusion of these symbols is not a definitive decision. As it will be explained in the future work section of this paper, we intend to make our level oriented methodology undergo a thorough testing procedure. One of the main goals of these tests would be to determine the suitability of the selected symbol set for the modeling of all types of business processes. With the results obtained from the tests we will be able to determine the need to include additional symbols to the methodology or discard the inclusion of any other entity to our methodology.

7) Applying BPLOM to a business process

The initial level of BPLOM has been used to model real life business processes at a Spanish enterprise called Isastur. The business processes that were modeled represented to areas of the enterprise with different characteristics: one was the informatics incidence management process and the other was the recruitment process.

This difference in characteristics allowed BPMN MUSIM to be considered suitable for modeling different kinds of processes and at this point we are going to use BPLOM to represent a model from a completely different nature. In this case we are going to use our level oriented approach to illustrate a product catalog application, including the possibility of buying the goods at the en of the process. The different figures in this section will represent the aspect of the process model as it passes through three of the levels in BPLOM: level 0, level 2 and level 4. These levels have been chosen because they correspond to the initial, middle and last stages of the methodology.

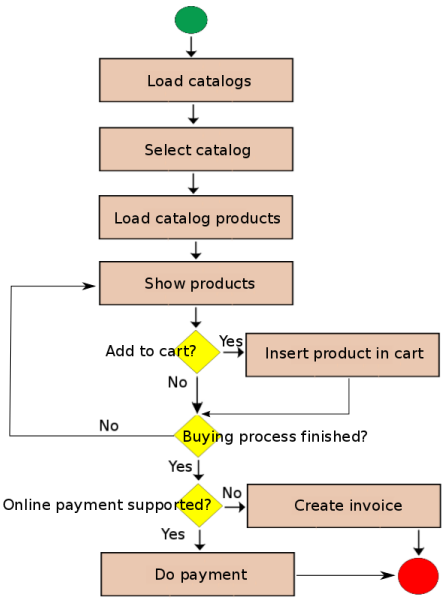

Fig. 16. Level 0 catalog application process model

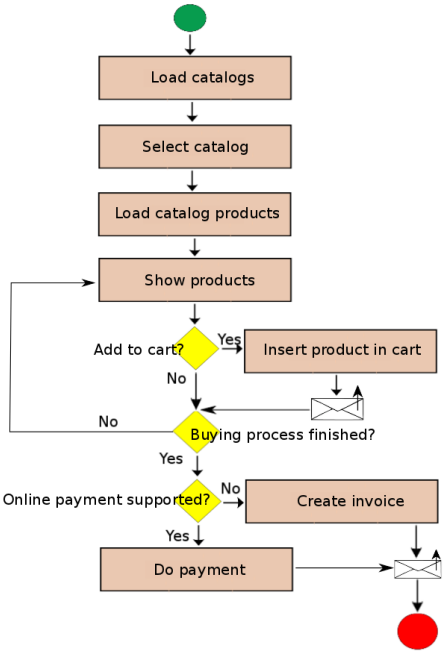

Fig. 17. Level 2 catalog application process model

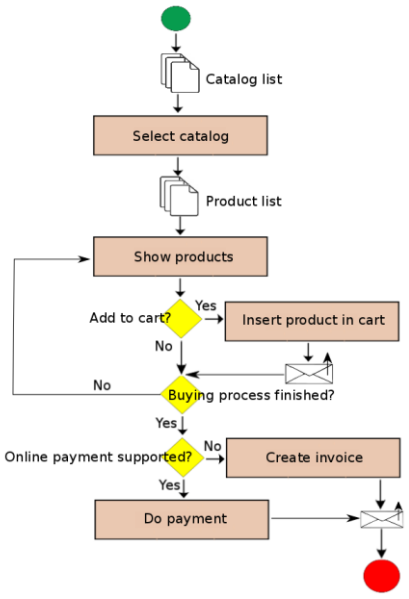

Fig. 18. Level 4 catalog application process model

Looking at the three precedent figures the evolution a process undergoes when passing through the levels of our methodology can be clearly seen. A BPMN MUSIM version of the process is shown in Fig. 16. As it can be seen this process contains no technical detail at all, as this level of our methodology is targeted towards non-technical application domain experts. Once the domain expert has gained some experience with the initial level of the methodology, he would start going up through the following levels, and thus an increase in the number of entities available for modeling and of the notation expressiveness would take place.

A non-technical expert with medium modeling skills would have access to level 2 of the methodology and therefore would design the process seen in Fig. 17. In this case, the inclusion of the outgoing messaging events gives the process additional functionality and shows the richer expressiveness of the notation at this level.

The last process, shown in Fig. 18, represents the use a business process modeling expert with some technical knowledge would give to the methodology. For this particular process the difference between the process models built with level 2 and 4 of the methodology is the inclusion of the object collection entities. This circumstance shows that the business expert that modeled the process has some technical knowledge as he manages to understand the concept of an object collection.

IV. BPLOM Tools: BPLevel Modeler and BPLevel Generator

As it was mentioned in the features section, BPLOM enables the definition and execution of the processes modeled with the proposed entity set. In order to achieve this functionality, BPLOM requires the development of a couple of prototypes that enable the digital definition and transformation of the models. These prototypes have been called BPLevel Modeler and BPLevel Generator.

Despite the platform independent nature of the proposed methodology both prototypes have been built using Windows Workflow Foundation. This is due to the degree of customization that this platform offers for creating a modeling tool like the one that will be introduced. The fact of using this platform for creating our modeling tool has no effect on the platform independency feature shown by the methodology, as it will be explained later in this section.

A. BPLevel Modeler: Graphical Definition of BPLOM Models

BPLevel Modeler is a business process-modeling tool that supports the BPM level methodology proposed in this paper. This tool has been developed under the .NET platform, as Windows Workflow Foundation [22] offers an attractive scenario for developing highly configurable business process modeling tools.

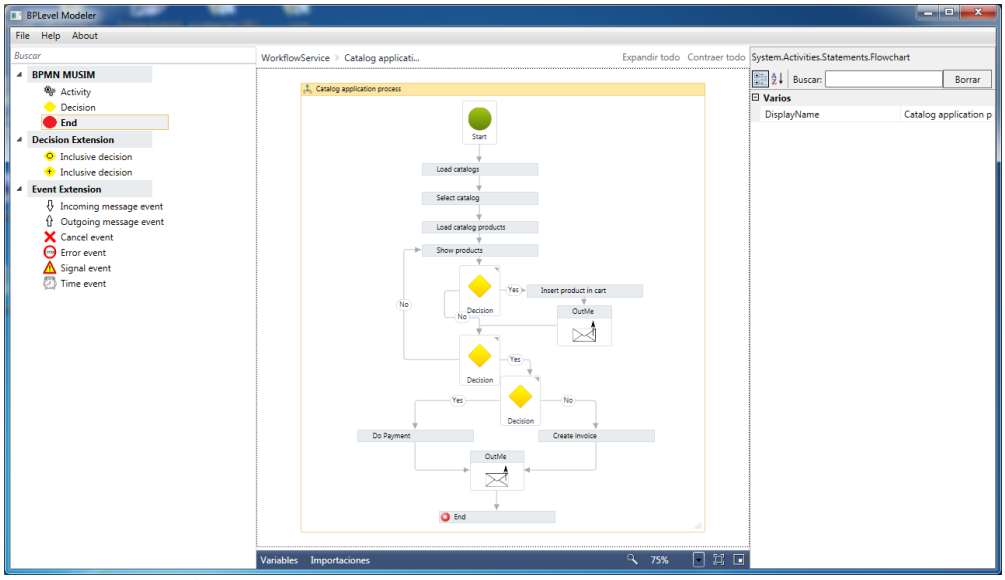

Fig. 19. BPLevel Modeler graphical user interface

The graphical user interface of BPLevel Modeler is shown in Fig. 19. As it can be seen, BPLevel Modeler has been kept as simple as possible in order to adapt to the computer skills shown by non-technical business experts. This tool has only one screen, which was divided into three different areas:

- The entity section, on the left of the screen, contains the different entities that can be used during process modeling. This section has been divided into five different categories, which represent the five levels of the methodology.
- The modeling section takes the center of the interface and is intended for the definition of the business process. It is closely related to the entity section, as the components in it can be dragged into the modeling section to define the desired business process.
- The property section, situated at the right hand side of the tool's screen, contains the property view of the BPLOM elements. The contents of this section depend on the element selected in the modeling section.

As BPLOM is designed as a skill level adaptive methodology, BPLevel Modeler must also adapt to the skill level presented by the user. In order to do so, when the tool is executed it will ask the user his knowledge level and based on the expert's choice the entity section will be adapted to show only the corresponding elements. Fig. 19 shows an entity section corresponding to a Level 3 expert and the business process model for the catalog application for that level.

Although BPLevel Modeler has been developed with Windows Workflow Foundation, which is included inside the .NET platform, the models it generates are considered platform independent. Windows Workflow Foundation stores models in a XML enriched format named XAML, a normal XML file with additional information regarding the position of the elements in the graphical representation of the process. Thus, BPLevel Modeler archives could be transformed using platform independent artifacts (like XSLT stylesheets, for example) and therefore used to generate code for any desired platform. As a matter of fact, in the use case described in this paper BPLevel Modeler files will be transformed using BPLevel Generator to generate mobile device applications for the Android and iOS platforms.

B. BPLevel Generator: Creating custom apps for BPLOM Models

BPLevel Generator is our code generation application. It analyses business process files built with BPLevel Modeler and generates multiplatform applications with specific characteristics for the given BPLOM model. Despite the fact that it is intended for computer experts this code generating tool has also been kept as simple as possible.

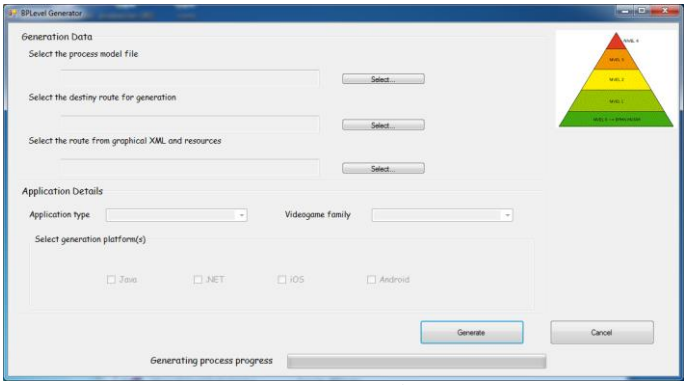

Fig. 20. BPLevel Generator graphical user interface

Fig. 20 shows a screen capture of the BPLevel Generator interface. As it can be seen, this tool requires the user to introduce several pieces of information:

1. The BPLevel Modeler file that represents the process that is going to be used as the basis for the code generation.
2. The path where the application(s) resulting from the generation process will be stored.
3. The path where the XML file with the graphical information of the application and the media resources are stored.
4. The platform(s) that the software is going to be generated for.

It must be pointed out that at this moment not all the generating features of BPLevel Generator are functional, but it is prepared for the inclusion of this functionality as a result of the planned future work.

Once the user introduces this data and starts the generation process, BPLevel Generator begins the analysis of the business model provided by the user and transforms this into the application represented by the details provided by the user. The generation process is divided into the following steps:

- **Preparing the creation structure.** Based on the path provided by the user as destiny for the generation process, BPLevel Modeler prepares the route for receiving the generated app.
- **Creating the custom app.** Code templates of the application(s) for the desired platform(s) are placed inside the path provided as the destiny of the generation process.
- **Configuring the GUI of the application.** BPLevel Generator analyses the XML file containing the graphical details of the application and substitutes the values of these details in the corresponding code class inside the application. BPLevel Generator also copies all of the resources the application needs into the appropriate folder inside the application.
- **Configuring functionality of the application.** Through an analysis of the business process model created with BPLevel Modeler our code generating tool substitutes the needed lines of code inside the application.

Fig. 21. Overview of the phases in the generating process and its result in the application's code for the background image of the main screen

Fig. 21 shows an example of how BPLevel Generator manages to substitute the graphical configuration information inside the application that is being generated. As it can be seen, the XML file contains three types of details for each of the controls or elements inside a given screen of the application: position (x and y coordinates), size (width and height values) and image. For each of the elements these values are stored in a specific class inside the application and then the instances of the elements in each screen are created using these values. The figure shows as an example the graphical configuration of the background image used in the main screen of the iOS application, as the rest of the elements are configured in a very similar way.

V. Conclusions

Requirements engineering is a critical task in the software development process, as it is the first phase done during the construction of any IT application and it establishes the needs and characteristics of the software. Despite this importance, requirements engineering has seen little changes in the last years and difficulties stated by Sommerville and Kotonya in their 1996 work [1] remain valid: the ambiguous character of natural language, also stated by Laue and Gadatsch [11], and the fact that the requirements engineer is not an expert of the application's domain. With these difficulties, another approach for the requirements engineering process arose: business process modeling. This discipline promotes the implication of application domain experts through the use of notations half way between computer knowledge and business domains.

Several business process modeling notations have appeared since BPMN was defined, in some cases trying to reduce the complexity level demonstrated by BPMN. This is the case of SBPMN [6][20], a simple business process modeling notation previously defined at the University of Oviedo. The mentioned complexity rate makes some of the notations difficult to use and understand by application domain experts with little or none modeling and technical knowledge. In addition to the complexity for modeling processes with these notations, several studies on the usage rates of the symbols in BPMN [3][7][19] have demonstrated the low use of some of the symbols included in these notations.

Bearing in mind the figures of these symbol usage studies and the difficulties found by business experts due to the complexity of these notations, we have tried to define a business process modeling methodology with two main goals: adapt the notation's complexity to the modeling and technical knowledge of the business experts and reduce the symbol set available in accordance with the mentioned usage rates. In order to adapt the notation to the expert's conditions, our methodology promotes the definition of five incremental levels, which add symbols to the notation gradually. When using the last level in our methodology, the expert is capable of using 20 symbols for modeling its processes; with this number of symbols, we manage to minimize the loss in expressiveness in comparison to BPMN and, at the same time, make our methodology capable of modeling any type of business process.

Once we defined this level oriented business process methodology, we decided to create two tools to take advantage of the methodology's features: platform independency and adaptive complexity. In first placed we built BPLevel Modeler, our business process modeling tool with support for our incremental level approach. Depending on the level selected by the business expert at the startup of BPLevel Modeler, the tool will automatically adapt its entity section to hold only the symbols available in the selected level; for example, if the user indicates its knowledge level is Level 2, BPLevel Modeler will show in its entity section all the symbols in levels 0, 1 and 2.

The other tool we have created is named BPLevel Generator, a code generating tool intended for IT experts. As business process models created with BPLevel Modeler are stored in an extended XML format and our methodology is platform independent, with BPLevel Generator we are capable of generating custom applications for each business process model constructed. For this generation to take place, BPLevel Generator needs the following information: the business process model file, the XML file with the graphical definition of the user interfaces the application has and the resource files used, the route for the code to be stored in and the deployment platform (Android or iOS). With these pieces of information,

BPLevel Generator is capable of analyzing the business process created with BPLevel Modeler in order to create a custom application with support for that specific process model in the desired platform, taking advantage of the platform independency characteristic of our level oriented methodology.

With this approach we manage to achieve the goals presented at the beginning of this paper. The first objective was to create a modeling notation capable of adapting its complexity to the skill level shown by the user, goal achieved through the definition of the level oriented methodology that fosters the adaptation of the modeling notation to the skills and knowledge the business expert has. We were also keen on involving business experts in the requirements engineering phase; this objective was accomplished with the creation of BPLevel Modeler, a simple, graphical business process modeling tool with support for the level oriented basis of our methodology. Lastly, we intended to use the generated models to generate the code of the custom applications that would give support to this models; with BPLevel Generator we can analyze the models created by the business experts and generate the mobile applications that represent them.

VI. Future Work

This paper includes some usability results that were carried out in order to establish the simplicity of BPLOM Level 0 and the completeness of its symbol set. In order to test the entire methodology we need to carry out some further testing with the rest of the proposed levels, the first point in our future work list. These usability tests will be focused in measuring the following aspects: the entities' graphical representation in comparison with BPMN, as this methodology intends to increase the experts' usability and abstraction levels; and the suitability of each of the levels regarding the entities they include and the order they are presented in, intending to minimize the time needed for the expert to advance through the levels. In this extended testing procedure we will also be able to determine the need to include more symbols present in BPMN to our notation, as it was already said in part six of the third section of this paper.

Another important point in our future work schedule is to change the way in which the XML file with the app's graphical information is created. At this time, the IT expert needs to manually introduce the details included in this document; these details include image files for the backgrounds, positioning of the elements inside the screen and size of these elements. Our main goal is to design and program a graphical user interface that enables the business expert to define the appearance of the different screens that make up the app easily. We are considering a drag and drop approach where the business expert can include backgrounds, buttons and other controls inside a canvas and adapt their look and feel to the style he prefers.

As it could be seen in Fig. 1, the proposed business process methodology is used to generate multiplatform mobile apps. In the use case documented in this paper, the generated software is a catalog app that allows users to scan through product catalogs, add them to their basket and buy them. At this moment the apps are generated for the most widely used platforms for this type of devices: Android and iOS. Although these platforms allow us to reach to the majority of mobile device users, we are interested in offering support not only for other mobile device platforms like Windows Phone but also for desktop and web applications through other platforms like Java and .NET. Moreover, we are also attracted by the possibility of creating other types of applications for the currently supported mobile platforms; for example, BPLOM could be used to model the process behind multimedia mobile apps like interactive books or even videogames, allowing the corresponding business experts to design and generate their own applications.

References

[1] Kotonya, G. and Sommerville, I. "Requirements engineering with viewpoints". 1996. *BCS/IEE. Software Eng. J.* , 11(1): 5-18

[2] Stephen A. White, "Introduction to BPMN". Business Process Management Iniciative (year 2004).

[3] Michael zur Muehlen, Jan Recker. "How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation". 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008).

[4] Lauri Eloranta, Eero Kallio y Ilkka Terho. "A Notation Evaluation of BPMN and UML Activity Diagrams" (year 2006).

[5] T.Wahl and G. Sindre. "An Analytical Evaluation of BPMN Using a Semiotic Quality Framework". CAiSE'05 Workshops. Volume 1, pages 533–544. FEUP, Porto, Portugal (year 2005).

[6] Fernández, H. F., et al. "SBPMN – An easier business process modeling notation for business users". Computer Standards & Interfaces 32 (1-2):18–28.

[7] Jan C. Recker. "BPMN Modeling – Who, Where, How and Why". BPTrends (year 2008).

[8] Stephen A. White y Derek Miers. "BPMN Modeling and Reference Guide: Understanding and Using BPMN". Future Strategies Inc. (year 2008).

[9] Ryan K. L. Ko. "A computer scientist's introductory guide to business process management (BPM)". Crossroads v.15 n.4, p.11-18 (year 2009).

[10] R. Lu y S. Sadiq. "A Survey on Comparative Modelling Approaches". Proc. BIS'07 (year 2007).

[11] Ralf Laue y Andreas Gadatsch. "Measuring the Understandability of Business Process Models - Are We Asking the Right Questions?". Business Process Management Workshops - BPM 2010 International Workshops and Education Track (year 2010).

[12] Jan Recker, Niz Safrudin y Michael Rosemann. "How Novices Model Business Processes". Business Process Management – 8th International Conference (year 2010).

[13] Zhiqiang Yan, Hajo A. Reijers y Remco M. Dijkman. "An evaluation of BPMN Modeling Tools". Business Process Modeling Notation - Second International Workshop (year 2010).

[14] C.A. Petri. "Kommunikation mit Automaten". PhD thesis, Institut fiir instrumentelle Mathematik, Bonn. (year 1962).

[15] W.M.P van der Aalst et al. "The Application of Petri Nets to Workflow Management". The Journal of Circuits, Systems and Computers, 8(1):21–66. (year 1998).

[16] Jaime Solís Martínez, Vicente García Díaz, Begoña Cristina Pelayo García-Bustelo y Juan Manuel Cueva Lovelle. "BPMN MUSIM: Notación BPMN muy simplificada". 6ª Conferencia Ibérica de Sistemas y Tecnologías de Información (CISTI 2011).

[17] Jaime Solís Martínez, Vicente García Díaz, Begoña Cristina Pelayo García-Bustelo y Juan Manuel Cueva Lovelle. "Isastur Modeler: A tool for BPMN MUSIM". 6ª Conferencia Ibérica de Sistemas y Tecnologías de Información (CISTI 2011).

[18] Michele Chinosi and Alberto Trombetta. "A design methodology for BPMN". Chapter in the 2009 BPM and Workflow Handbook. (year 2009).

[19] Michele Chinosi, and Alberto Trombetta. "BPMN: An introduction to the standard." *Computer Standards & Interfaces* 34.1 (year 2012).

[20] H. Fernandez-Fernandez, E. Palacios-González, V. García-Díaz, B. Cristina Pelayo G-Bustelo, Oscar Sanjuán Martínez and Juan Manuel Cueva Lovelle. "Developing a Business Application with BPM and MDE". International Journal of Artificial Intelligence and Interactive Multimedia (IJIMAI). (year 2009).

[21] A.P. Castaño. "Prototype of assignment intelligent adaptive of service providers inside of ESB with data mining". International Journal of Artificial Intelligence and Interactive Multimedia (IJIMAI). (year 2009).

[22] Windows Workflow Foundation. Last visit on May 5th 2013. http://msdn.microsoft.com/es-es/netframework/aa663328

[23] OMG. Last visit on May 6th 2013. http://www.omg.org/

[24] BPMI. Last visit on May 7th 2013. http://www.omg.org/bpmi

[25] BPMN. Last visit on May 5th 2013. Last update on April 29th 2013. http://www.omg.org/bpmn/

[26] UML. Last visit on May 4th 2013. Last update on April 15th 2013. http://www.uml.org/

[27] jBoss jBPM. Last visit on May 4th 2013. http://www.jboss.org/jbpm

[28] MDE Research Group in the Department of Computer Science, University of Oviedo. Last visit on May 7th 2013. Last update on December 2012. https://sites.google.com/site/mdeootlab/Home

Jaime Solís-Martínez is a Ph.D student in the Computer Science Department of the University of Oviedo. He has a B. Sc. in Computer Science Engineering and a M. Sc. in Web Engineering. His research interests include Model Driven Engineering, Domain Specific Languages, Business Process Modeling and the application of these technologies to web and mobile device applications.

Natalia García-Menéndez has a B. Sc. in Computer Science Engineering and a M. Sc. in Web Engineering, both at the University of Oviedo. Currently serving as a multiplatform mobile device programmer in the GADE4ALL project, her research interests include DSL, MDA, programming of visual editors for application design and using all these technologies in mobile device apps.

B. Cristina Pelayo G-Bustelo is a Lecturer in the Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering. Her research interests include Object-Oriented technology,Web Engineering, eGovernment, Modeling Software with BPM, DSL and MDA.

Juan Manuel Cueva Lovelle is a Mining Engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). Ph. D. from Madrid Polytechnic University, Spain (1990). From 1985 he is a Professor at the Languages and Computers Systems Area in Oviedo University (Spain). ACM and IEEE voting member. His research interests include Object-Oriented technology, Language Processors, Human-Computer Interface, Web Engineering, Modeling Software with BPM, DSL and MDA.