A Method-wise Approach for Selecting the Most Suitable Business Process Modelling Notation

Gianna Reggio, Maurizio Leotta

Abstract:

When modellers have to describe a business process, they can decide to rely on one of the several notations currently available such as for instance BPMN or UML. However, it is not easy to select the most appropriate one. Indeed, even though in the literature several comparisons among notations have been proposed, often they focus on specific properties of the notations or on their constructs. On the contrary, the real concern of the modeller is to select the best modelling method, based on a notation they know for a specific modelling case.

For this reasons, in this work we further raise the level of comparison by proposing an approach based on matching the features, required by the modelling case that the modeller has to face, with the ones supported by available modelling methods based on those notations. In this way the modeller can really select the best method (based on a modelling notation) according to the needs of the specific modelling case.

Digital Object Identifier (DOI):

https://doi.org/10.1109/SEAA.2019.00068

A Method-wise Approach for Selecting the Most Suitable Business Process Modelling Notation

Gianna Reggio, Maurizio Leotta
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università di Genova, Italy
gianna.reggio@unige.it, maurizio.leotta@unige.it

***Abstract*—When modellers have to describe a business process, they can decide to rely on one of the several notations currently available such as for instance BPMN or UML. However, it is not easy to select the most appropriate one. Indeed, even though in the literature several comparisons among notations have been proposed, often they focus on specific properties of the notations or on their constructs. On the contrary, the real concern of the modeller is to select the best modelling method, based on a notation they know for a specific modelling case.**

For this reasons, in this work we further raise the level of comparison by proposing an approach based on matching the features, required by the modelling case that the modeller has to face, with the ones supported by available modelling methods based on those notations. In this way the modeller can really select the best method (based on a modelling notation) according to the needs of the specific modelling case.

***Index Terms*—Business process modelling, BPMN, UML, Modelling Method.**

I. INTRODUCTION

Currently many notations are available for the business process modelling: BPMN [1], UML activity diagrams [2], EPC [3] and variants of coloured Petri nets [4] are among the most known. As a consequence, many scientific investigations (see e.g. [5]–[12]) consider the relationships between such notations and their mutual comparison to help modellers to select a sensible notation for their specific modelling cases. These investigations [5]–[12] (1) often concern general properties of the notations (e.g. understandability, usability, expressivity); (2) try to establish the relationships among the constructs of the notations; or (3) propose to identify the constituent elements of the business process, and then to relate them with the notations' constructs. In few cases the reasons motivating the production of the models are considered (e.g. for documenting or for generating automatically some software artefacts) to drive the choice of the proper notation. As stated by Aguilar-Savén [12]: "*It is important to identify the uses or purposes of the models when undertaking modelling of any kind. [. . .] Different techniques are more suitable to certain purposes, e.g. one thing is a model, which describes the process, and another a model to build a system to control the process.*"

However, we think that it could be more fruitful to further raise the level of comparison, i.e. not just to compare *modelling notations* but instead *modelling methods* that use some notations. Indeed, given a business process BP and a modelling notation N there is not a unique way to model BP using N.

According to Reggio et al. [13] a *modelling method*: (1) is based on one *modelling notation*, (2) provides precise indications of what are the *modelled items* (e.g. data-aware business processes), (3) defines the subset of the models of the chosen notation used by the method, the *eligible models* (e.g. the BPMN models consisting of just one collaboration diagram), (4) expresses the relationship between the modelled items and the eligible models (the *modelling essence*), (5) gives *guidelines* to drive the modeller activity, and (6) explicitly details the *intended use* of the produced models.

Thus, when modellers have to face a modelling case, they will not look for the best notation, but for the best method based on a notation they know and according to a set of characteristics specific of the modelling case to complete.

To the best of our knowledge, no one has proposed to relate *modelling methods* for the business processes. In this paper we propose a quite comprehensive set of features of business processes and of their modelling that may influence the choice of a proper modelling method. Such features are emerged by our experience in modelling business processes, see [14] and in our research activities on this topic [15]–[19] while others are new.

Each specific case of business process modelling should allow to determine the required features, thus characterizing the modelling task to undertake. Then, two given business process modelling notations (say A and B) should not be compared construct by construct or with respect to generic characteristics (e.g. understandability), but with respect to sets of features characterizing modelling cases, i.e. is it possible to work out a modelling method supporting a given set of features and based on A/B?

Adhering to a method-wise way to compare modelling notations may lead to results rather different from those of other approaches reported in the literature, but clearly really useful to the modellers. Consider for example the case of the *business process goals* feature (i.e. the process goals are required in the considered modelling case): the classic answer is that neither BPMN nor UML may cope with the goals; however, in the case of UML, by using stereotyped use cases or classes, it is possible to model the process goals and their decomposition in sub-goals.

Summarising, the novelties of the proposed approach for comparing modelling notations are:

- the approach takes into account the characteristics of the specific modelling case the modeller has to face;
- the result of the comparison is a modelling method based on a notation: thus the modeller is ready to start to model and does not need to think how to use the "best" notation on their specific case;
- the application of the approach to several modelling cases results in one catalogue of modelling methods and one of mappings *modelling case* ⇔ *fit method* that may be shared and reused.

In this paper, we consider only two notations for business process modelling, precisely BPMN 2.0 [1] (the most known and used) and UML 2.5.1 [2] (in our opinion wrongly considered unfit for real applications), and "compare" them with respect to some specific modelling cases, each one characterized by the required features.

In Sect. II we introduce our holistic conceptual view of business processes and of their models to provide means to define the characterizing features. Sect. III presents the selected features, and then in Sect. IV we compare method-wise BPMN 2.0 and UML 2.5.1 on several, in our experience quite common, modelling cases. Sect. V and VI present the related work, the conclusions and the future work.

II. A Conceptual View of Business Processes

In this section we first propose a holistic conceptual view of business processes, and then an analogous holistic view of their models with the aim of providing a basis for defining the features of interest when modelling business processes. Obviously, both are fully independent from the notation used to represent such models.

To make our explanation clearer we will use the following running example.

Buying process. *A dealer may place an order to a manufacturer to buy some quantity of a specific product e.g. gasoline. When a manufacturer receives an order, it will check the product availability. If the ordered amount of product is available, the order will be confirmed, otherwise, it will be cancelled. The dealer tries to pay for the order. Once the order is paid, the manufacturer will send a confirmation to the dealer, and will ask a shipper to deliver the product to the dealer. If the payment fails, the manufacturer will cancel the order. Once the shipment is delivered, the shipper will send a confirmation to the manufacturer.*

Looking for a holistic view of business processes leads to consider also the enterprise (or business) context in which the processes execute. We consider an *enterprise* as whatever organization, formally or informally established, that performs specific activities for specific *goals* (not only monetary profit); some samples are: companies producing material goods, banks, and universities.

We summarize our view of enterprises and business processes at a conceptual level in Fig. 1 by means of a UML class diagram (in this paper we follow the convention that in a UML class diagram the multiplicity of an association will be omitted whenever it is equal to 1), and below briefly describe their constituent elements.

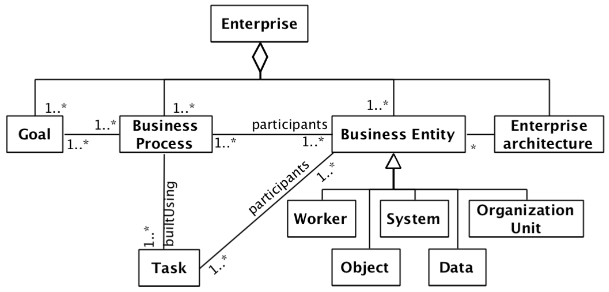


Fig. 1. Enterprise and Business processes Conceptual View

In an enterprise/business process there are a number of entities that perform activities manipulating other passive entities, that we call *business entities*. We classify the business entities in:

- *workers*, which are humans performing activities in the processes (e.g. a clerk, a buyer, a member of an association);
- *systems*, which are software or hardware systems performing automatically activities (e.g. an ATM, Paypal);
- *business objects*, which are digital and physical things being handled in the processes (e.g. an account, an invoice, a package, a pair of shoes, if the enterprise is a shoe factory);
- *data*, which are just plain data handled in the processes (e.g. credit card numbers, amounts of money);
- *organization units*, which are the logical units defining the organization of the enterprise enacting the processes (e.g. departments, branches and sub-branches, plants); notice that the organization units cannot perform autonomous activities as the workers and the systems, but they can be modified (e.g. a branch is closed). Organization units may be hierarchically decomposed into other organization units and may encompass also business entities of other kinds, thus they define the organizational structure of the enterprises.

Workers and systems are termed *active entities*.

A *business process* is motivated by some goals, and is characterized by some roles to be played by business entities, the process's *participants* distinguished in "in" (they need to be filled to start the process, e.g. a student enrolling in a degree), and "out" (they will be *filled* by the process, e.g. a shipper role that will be filled during the process execution by choosing among a set of registered shippers).

A business process is built out of a set of basic activities (*tasks*) organized by causal and temporal relationships (sequencing, parallelism, and conditional choice); each task is in turn characterized by some participants, i.e. roles for the business entities taking part in it. Tasks may be classified as – *abstract*, they have any number of participants (obviously at least one active) and no one of them has a prominent or principal role, and – *executable*, they have either one or two active participants (representing an activity made by one participant by itself and a communication from one participant towards another one, respectively).

Abstract tasks may correspond to complex activities taking a lot of time to be executed, but they are considered atomic when modelling the business processes, thus the models may be quite abstract, whereas executable tasks oblige to view the process at the execution level and as a result at a lower level of abstraction. For example, "the student, the supervisor and the head of the faculty decide the date of the final exam" (three workers) is an abstract tasks, using executable tasks it must be decomposed in smaller tasks, such as "the supervisor proposes a date to the head of the faculty", "the head of the faculty agrees", and finally "the supervisor informs the student of the chosen date". Both the abstract business processes (i.e. made of abstract tasks) and the executable ones (i.e. made of executable tasks) are useful. The former may present complex administrative processes without irrelevant details, and the latter processes to be supported by a software system.

In this paper we do not detail how the tasks are put together to define the process workflow, since the flow control constructs have been extensively studied, and because BPMN and UML provide a huge number of constructs of this kind (e.g. 49 constructs for the UML activity diagrams), and so the flow constructs are not a key feature to select the modelling notation.

The elements of an *enterprise* are: its goals, its business entities, its *architecture* that shows how and which business entities build the possible configurations of the enterprise, and the processes that it enacts.

For example, referring to the business community enacting the Buying process we have an enterprise with:

- a unique goal: "make as many deals as possible";
- many different types of business entities: dealer, manufacturer and shipper (worker), electronic payment (system), and order (object); since the organization of this enterprise is very simple there are no organization units;
- a business process, Buying;
- some sample tasks of the process Buying are: "a manufacturer sends a shipping request to a shipper", and "a manufacturer informs a dealer that an order has been cancelled".

An *enterprise model*, see Fig. 2 is made of a *goal view*, an *entity view* (introducing and modelling the entities parts of the enterprise), a *business process view* (introducing and modelling the business processes enacted by the enterprise), and a representation of the *enterprise architecture*.

An entity view is a collections of business *entity models* (obviously their forms will depend on the kind of the modelled entities). However, each entity model should both define the static structure and the dynamic behaviour of the modelled entity (e.g. for entity of kind business object the structure will be the form of their possible states and the behaviour will define how they may pass from one state to another).

A *business process view* consists of a *process overview* that summarizes all the processes enacted by the enterprise possibly with their mutual relationships and participants, plus a collection of business process models. A *business process model* consists of a set of *participants* (roles typed by business entities), a mandatory *workflow view* built out of the tasks (whose behaviour is modelled separately), and optionally by some *interaction views* depicting the behaviour of the process in term of message exchanges among its active participants.

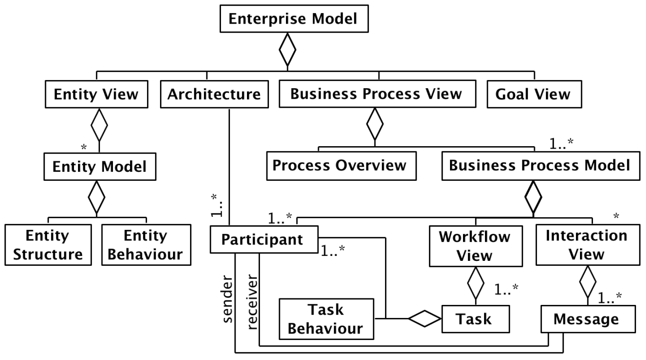


Fig. 2. Enterprise and Business process Models Conceptual View

III. Features of Business Process Modelling

The conceptual views of business processes and their models introduced in Sect. II allowed to extract a set of features. The considered features are:

- *elements* of the business processes (e.g. participants of kind data, goals),
- *views/parts* of the models (e.g. the interaction view of the business processes, the definition of the data),
- *dimensions* of the business processes defined by means of metrics (e.g. number of participants/tasks of the business processes),
- *characteristics* of the models (e.g. degree of formality); here we consider only "objective" characteristics, that are properties of the models that can be verified without the human intervention. For example readability is not objective (indeed it can only be verified by means of empirical assessments with human participants), whereas minimality is objective, remind that a model is minimal iff all its parts are necessary, and thus if one is removed the model becomes ill-formed. Non-objective characteristics have been extensively studied and many results are reported in the literature.

Notice that the considered features have not been inspired by any specific notation neither by any specific modelling construct, they try to abstractly characterize, as much as possible, business processes and their modelling.

The first column of Table I summarizes all the features considered in this paper.

Elements and Views: the constituent elements of business processes and of enterprises introduced in Sect. II have prompted the definition of the features of kind element, those marked by (E) in the first column of Table I, while, the parts/ views of the business process/enterprise models introduced again in Sect. II have prompted the features marked by (V).

Dimensions: different modelling methods may cope differently with "large" business processes, thus it is important to consider also the dimensions of the modelled items to select an appropriate modelling method.

The relevant dimensions in this case are the number of tasks and of the participants. A great number of tasks leads to workflow views being large and complex graphs, that need to be modularized, whereas a great number of participants will result in cluttered workflows depending on the means used to represent them; think for example the case of active

TABLE I
BPMN/UML BASED MODELLING METHODS WITH SUPPORTED FEATURES

	Methods						
	BPMN		UML				
Features	**B1**	**B2**	**U1**	**U2**	**U3**	**U4**	**U5**
Participants (E)	✓	✓	✓	✓	✓	✓	✓
Active participants (E)	✓	✓	✓	✓	✓	✓	✓
Workers & Systems (E)	P	P	✓	✓	✓	✓	✓
Active Participant Structure & Behaviour (V)	☒	☒					✓
Business Objects (E)	✓	✓	✓	✓	✓	✓	✓
Business Object Structure & Behaviour (V)	☒	☒	✓		✓		✓
Data (E)	✓	✓	✓	✓	✓	✓	✓
Data Definition (V)	☒	☒	✓		✓		✓
Organization Units (E)							✓
Tasks (E)	✓	✓	✓	✓	✓	✓	✓
Task Participants (E)	✓	✓	✓	✓	✓	✓	✓
Task Behaviour (V)	☒	☒	✓		✓		✓
Abstract tasks (E)	P		✓	✓			
Executable tasks (E)	P	✓			✓	✓	
Workfow View (V)	✓	✓	✓	✓	✓	✓	✓
Interaction View (V)		✓					
Messages (E)		✓					
Goals (E)							✓
Multiple Processes (E)							✓
Process Overview (V)							✓
Entity View (V)			✓	✓	✓	✓	✓
Enterprise Architecture (E)							✓
Long Processes (D)	P	P	✓	✓	✓	✓	✓
Large Processes (D)	P	P	✓	✓	✓	✓	✓
Formal (C)	☒	☒					
Hard (C)	☒	☒	✓		✓		✓
Soft (C)	☒	☒		✓		✓	

✓ supported P partially supported ☒ cannot be covered by any extension of the method

Feature kind: **E** Element, **D** Dimension, **V** View, **C** Characteristic

participants rendered by swimlanes/lanes/pools with the flows crossing many lanes, or to the case when using UML each flowing of data and business object between tasks will be modelled by an object node.

We define a business process to be "long" iff the numbers of its tasks is larger than 20, indeed it will be difficult to examine using a standard computer screen or on a A4 page, and to be "large" iff either the numbers of its active participants is larger than 10 or the number of its non-active ones is larger than 10. Clearly, it is difficult to place exactly the thresholds, the figures used by our definitions are motivated by our practice in modelling also quite large processes, but they need to be validated by empirical experimentations. Silver [20] states that a process is large and thus must be decomposed when it cannot be drawn on an A4 page, and thus when the number of tasks is larger than 10; our thresholds correspond to values higher that those indicated by Silver, since we assume that models can be seen on a screen and not printed.

The features Long Processes and Large Processes determine the cases when long and large processes should be modelled (since the constructs used to model active and non-active participants are usually different it is possible that a modelling notation offers means to handle only one of the twos).

Characteristics: in this initial version, we consider only the formality related characteristics, in the future will plan to investigate which, if any, further objective characteristics may be added.

Visual models may be more or less "formal" going from drawings built using the icons and graphical elements provided by the chosen notation where the textual inscriptions are unconstrained natural language fragments, to models where the textual inscriptions are expressed using textual formal languages. Different modelling cases may require different levels of formality, for example models intended to be read by the users of an application to learn to use it do not need to be very formal. On the other hand, models that will be automatically translated into code without any further human intervention need to be quite formal.

We propose the following classification, a modelling method is: – Formal, if the used notation has a precisely defined syntax, static semantics, and a formal semantics (e.g. coloured Petri nets are formal); – Hard, if the used notation has a precisely defined syntax, static semantics, and semantics (e.g. Java is hard); – Soft, if the used notation has a precisely defined syntax and static semantics (e.g. any language defined by a XML schema is soft); – loose otherwise.

Note that different subsets of a notation may be classified differently, and also differently from the whole notation.

The visual constructs of both UML and BPMN are hard, but the differences with respect to this classification are given by the textual parts of their models. UML is quite flexible, indeed the textual parts may be expressed using OCL [21] and the UML actions with concrete syntax [2, sect. 16.1.1.1], and thus the models will be hard, but also using natural language text (e.g. using "opaque behaviour" [2, sect. 13.2.3.3] that is defined by natural languages fragments), and thus the models will be loose. So, it is possible to devise UML based methods that are loose, hard, and even formal using the fUML (www.omg.org/spec/FUML/1.1/About-FUML/) to provide UML with a formal semantics.

BPMN instead leaves underspecified the form of the textual parts, and so a modelling method based on it cannot be hard or even soft without overstepping the official specification of the notation.

IV. MODELLING USING BPMN AND UML

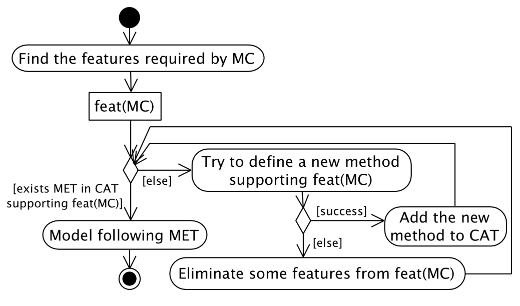
Using the features characterizing the business process modelling cases and methods introduced in Sect. III, we present our *method-wise* approach to modelling business processes using either BPMN 2.0 [1] or UML 2.5.1 [2] (the most recent versions), see Fig. 3. In this paper we consider only five, in our experience [14] quite common, modelling cases.

For any modelling case MC the goal is to select a method supporting all the features required by MC, say feat(MC). We assume to have a catalogue CAT of methods based on BPMN or on UML, accompanied by the list of the supported features.

The following cases are possible: a) a method covering feat(MC) belongs to CAT, then the goal is reached, b) no method in CAT supports feat(MC): b1) a new method covering feat(MC) may be designed and added to CAT, then the goal is reached, b2) otherwise some of the less relevant features should be removed from feat(MC), and the procedure should be repeated.

A. BPMN and UML modelling methods

Here we introduce two methods based on BPMN and five based on UML developed in the last years (see [14] for their complete definitions and applications). They have been prompted and then applied in joint projects with industries, and have been used and tuned by many students projects. All of them have been found fit for the modelling case motivating



Find the features required by MC
feat(MC)
[exists MET in CAT supporting feat(MC)]
[else]
Try to define a new method supporting feat(MC)
[success]
Add the new method to CAT
[else]
Model following MET
Eliminate some features from feat(MC)

Fig. 3. The method-wise approach to modelling

them, e.g. the soft UML models were easily understood by readers without a computer science background, whereas the hard ones were found suitable to be translated into inputs for an agent-based simulation tool and also in coloured Petri nets to be analysed using the CPN-tool (http://cpntools.org/).

To presents the methods we follow the schema of [13], and we define the eligible models precisely by a metamodel, defining the diagrams composing the models and their mutual relationships plus a set of well-formedness constraints. Furthermore, in each case the guidelines driving the modeller's work start with "try to find which are the business process elements that you have to model", for example which are the participants, and the tasks.

Table I lists the considered methods with the indication of which features they support. The methods are briefly presented in Sect. IV-B and IV-C, while their detailed definitions can be found in [14]. For each method we provide the model of the running example: the Buying business process.

B. BPMN based modelling methods

Active participants may be modelled in BPMN either using lanes or pools, in the second case the messages exchanged between two participants may be modelled. The method **B1** follows the idea "participants = lanes", on the converse **B2** adheres to "participants = pools".

• **B1** A business process is modelled by a BPMN process diagram enclosed in a pool, having a lane for each *active participant* named as the participant itself; as a consequence the tasks inside a lane will have a unique active participant (the one naming the pool). The distinction between workers and systems may be partly rendered by looking at the kind of the tasks included in their lanes: e.g. manual and user tasks imply worker, and script tasks imply system participant.

Data and business object participants are represented by `DataObject`s, and the fact that they participate in a task is represented by `DataAssociation`s (see [1, sect. 10.3.1]).

Well-formedness rules constrain the use of the various task types inside a lane (e.g. a lane cannot contain both manual tasks and script tasks), and further constraints regulate the use of in and out `DataObject`s.

B1 partly covers Abstract Tasks (only abstract tasks with one active participant), similarly it partly supports the Executable Tasks feature (only executable tasks with one active participant).

Long Processes is partly covered by **B1**, indeed BPMN provides the sub-process construct [1, sect 10.2.5], but the

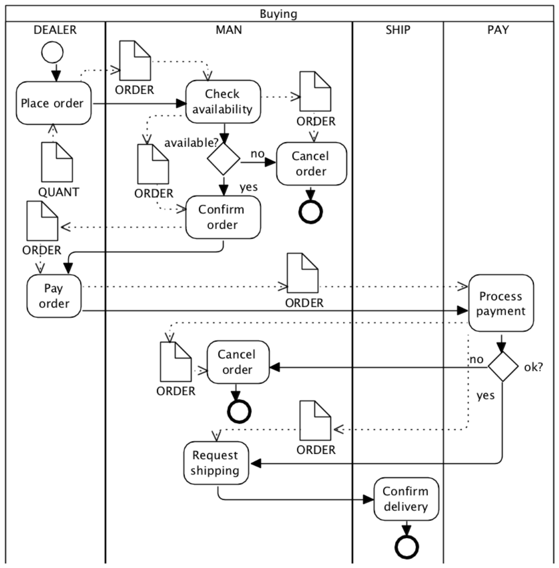


Fig. 4. **B1** model of the Buying process

sub-process is contained in one lane, thus only the parts of a process diagram contained in one lane may be modularised. Also Large Processes is partly covered, since the lane mechanism may render the diagram quite complex and difficult to produce and understand when the number of lanes is high and it is not possible to find an ordering of the lanes avoiding that the flows cross too many lanes.

B1 is loose, since the data and the `conditionExpression`s on the flows outgoing the gateways are just strings (indeed, nothing is said on how to express and how to insert then in the diagrams).

As explicitly stated in [1, sect. 7.1] there is no way to cope with Business Object Structure & Behaviour, and Data Definition, not even the active participant structure and behaviour in isolation may be modelled.

For what concerns the definition of the task behaviour, [1] only states that should be defined by an `InputOutputSpecification` associated with the BPMN task relating the received and the produced data. However, [1] does not say how to express the `InputOutputSpecification` and/or to represent the received and produced data in the models. Only for the script tasks it says that there is a script in some language associated with the task (but no indication on how to represent it in the model) that will be executed by some engine, and for the business rule tasks it hints that the `InputOutputSpecification` is determined by a business rule that will be computed by some external business rule engine, thus completely outside the model.

The **B1** model of the Buying process is reported in Fig. 4, note how the large number of `DataObject`s (to show completely the flowing of Order among the process tasks) clutter the diagram.

• **B2** is defined as **B1**, but now the active participants are modelled by pools, and thus it is possible to model the messages exchanged between the participants. A **B2** model consists of a collaboration diagram with one pool for each active participant.

B2 supports the feature Executable Tasks, indeed now a task with a unique participant will be a BPMN task inside a pool, a task with two participants corresponding to a communication among them will be modelled by a message exchange between two pools. **B2** does not support Abstract Tasks, think for example to the task of deciding the Ph.D. final exam date with three active participants cited before, it cannot be represented by a BPMN task inside a pool not even if such task sends several messages.

B2 partly copes with Large Processes and Long Processes as **B1**, since what said about lanes applies also to pools. The definition of the form of the **B2** models requires further well-formedness constraints (for example to avoid problems with the use of the messages flows). As **B1**, **B2** is loose.

A collaboration diagram provides both a Workflow View and an Interaction View. BPMN 2.0 provides also other ways to present simultaneously the Workflow View and the Interaction View, such as the choreographies. Thus, it is possible to extend **B2** with additional ways to present the interaction views; while the black-box pools (see [1, fig 7.6]) may be used to provide further additional interaction views, showing only the messages exchanged among the active participants while hiding the flowing of tasks inside each pool.

The **B2** model of the Buying process is reported in Fig. 5. The model is quite readable, but its understanding is totally based on the understanding of English language (e.g. translating only the sentence “available?” in a different language will disrupt the perceived meaning of the model).

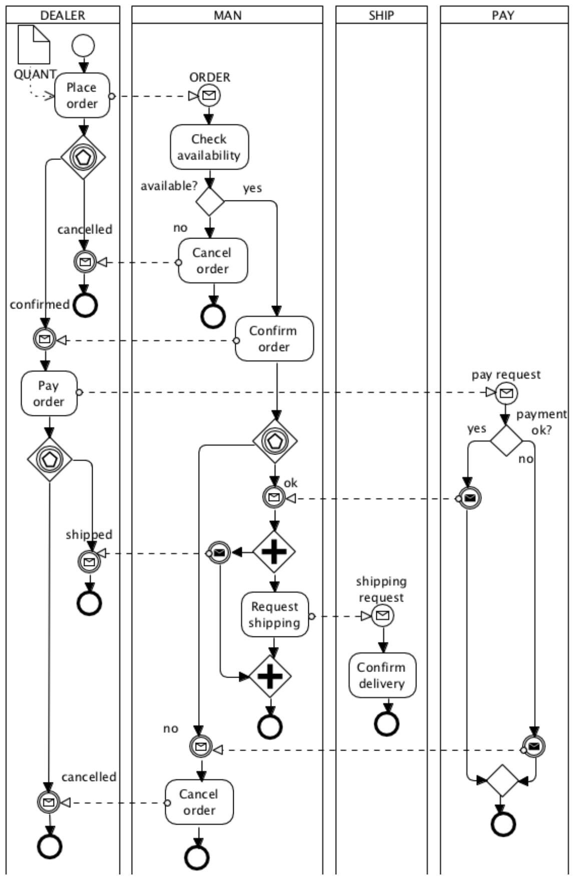


Fig. 5. **B2** model of the Buying process

C. UML modelling methods

The UML based methods considered in this paper, will take advantage of the many diagrams and constructs of UML, and will not limit themselves to propose models made of just an activity diagram; moreover, each method will use a specific UML profile defined by a set of stereotypes.

• **U1** Following **U1**, the types of active participants are modelled by UML active classes stereotyped by either ≪worker≫ or ≪system≫, the types of business objects by standard classes stereotyped by ≪business object≫ and their behaviour is given by methods or pre-post conditions associated with their operations, finally types of data are represented by UML data types and their definition is given again by methods or pre-post conditions. Then, the process participants, that are roles, are defined by pairs made by a name and a participant class. All classes and datatypes are collected in a class diagram (the Entity View).

The tasks will be instances of classes stereotyped by ≪task≫, whose attributes/connected associations define the task participants; and their behaviour will be defined either by pre-post conditions on the participants or by an associated detailed activity diagram. Thus, **U1** supports Abstract Tasks.

The workflow view consists of an activity diagram, such that:

– the action nodes are labelled by task instances, denoted by TaskClass<$E_1, \ldots, E_n$> where TaskClass is a task class and E_1, …, E_n are OCL expressions built using the participant names and the operations of the participant classes;

– the guards on the flows leaving the decision nodes are OCL expressions built using the participants names and the operations of the participant classes, and

– it does not contain swimlanes.

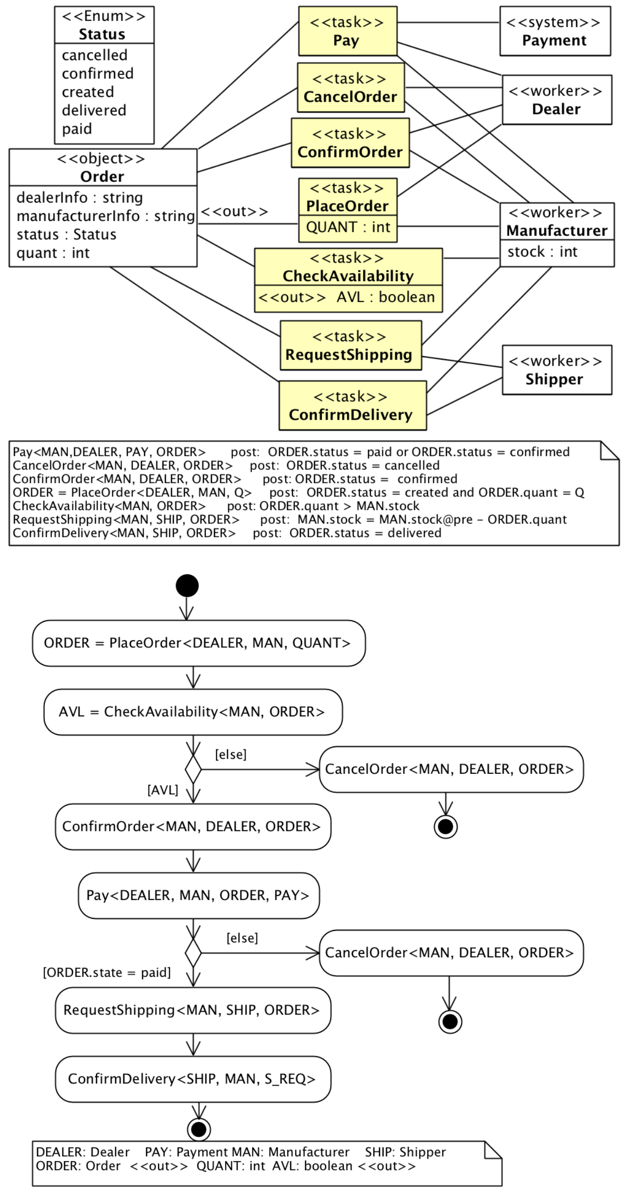
Using the sub-activity construct (the rake) it is possible to modularly decompose long processes, while a large number of participants will not affect the representation of the flowing of the activities, thus **U1** supports both Long Processes and Large Processes.

U1 is Hard, indeed both the OCL (used to define the task instances, the guards, and the pre-post conditions) and the UML actions with concrete syntax (used to define the methods) have a well-defined syntax, static semantics, and semantics.

The model of the Buying process made using **U1** is reported in Fig. 6, note how the payment activity is abstractly represented by the task Pay<MAN,DEALER,ORDER,PAY> with three active participants.

• **U2** is a slimmer version of **U1**, indeed now the participant classes have no attributes, and their operations are not defined at all, thus the Entity View will consists exactly in a set of classes without attributes and without methods/pre-post conditions associated with their operations; moreover, the task behaviour is not defined.

U2 is not Hard, indeed the semantics of the OCL expressions appearing in the activity diagram is not defined, because the operations used to build them are not specified.



```
Pay<MAN,DEALER, PAY, ORDER>      post:  ORDER.status = paid or ORDER.status = confirmed
CancelOrder<MAN, DEALER, ORDER>    post:  ORDER.status = cancelled
ConfirmOrder<MAN, DEALER, ORDER>     post: ORDER.status =  confirmed
ORDER = PlaceOrder<DEALER, MAN, Q>    post:  ORDER.status = created and ORDER.quant = Q
CheckAvailability<MAN, ORDER>     post: ORDER.quant > MAN.stock
RequestShipping<MAN, SHIP, ORDER>     post:  MAN.stock = MAN.stock@pre – ORDER.quant
ConfirmDelivery<MAN, SHIP, ORDER>    post:  ORDER.status = delivered
```

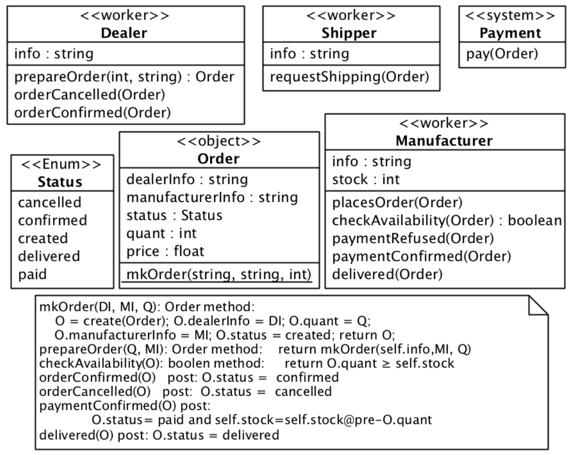
```
DEALER: Dealer   PAY: Payment MAN: Manufacturer   SHIP: Shipper
ORDER: Order  <<out>>  QUANT: int  AVL: boolean <<out>>
```

Fig. 6. **U1** model of the Buying process

The resulting models are easily readable by humans and the modelling effort is quite less than using **U1**, at the same time the inscription on the activity diagram are quite precise and so less prone to ambiguities (the Entity View will be in practice a set of declarations that will help to avoid to use different wordings for the same concepts, thus reducing the ambiguities, and will force to precisely state which are the participants of the tasks, and which operations will be made over data and business objects). Thus **U2** is Soft. Moreover, since it is easy to refine **U2** models into **U1** models **U2** may be used at the beginning of the development of **U1** models.

The model of Buying produced following **U2** is obtained by deleting the task pre-post conditions in the one in Fig. 6.

• **U3** is defined similarly to **U1**, but it supports Executable Tasks. The workflow view consists of an activity diagram with a swimlane for each active participant, named as the participant itself. An executable task is modelled as follows: a) task with



```
mkOrder(DI, MI, Q): Order method:
   O = create(Order); O.dealerInfo = DI; O.quant = Q;
   O.manufacturerInfo = MI; O.status = created; return O;
prepareOrder(Q, MI): Order method:    return mkOrder(self.info,MI, Q)
checkAvailability(O): boolen method:     return O.quant ≥ self.stock
orderConfirmed(O)   post: O.status =  confirmed
orderCancelled(O)   post:  O.status =  cancelled
paymentConfirmed(O) post:
            O.status= paid and self.stock=self.stock@pre−O.quant
delivered(O) post: O.status = delivered
```

Fig. 7. **U3** model of the Buying process: Entity View

a unique participant P (private activity of P) by P.op($E_1,\dots,E_n$) inserted in the lane of P, b) communication of participant P with P' by P'.op'($E_1,\dots,E_n$) placed in the lane of P, where op/op' is an operation of the class of P/P', $E_1, \dots, E_n$ are OCL expressions built using the process participant names. All operations of the participant classes must be defined by either associated methods or pre-post conditions. Since each task is an operation call, defining the meaning of the operations results in defining also the behaviour of the tasks.

The workflow view consists of an activity diagram whose actions are calls of participants operations, thus **U3** covers executable tasks.

An Interaction View is given by a sequence diagram with a lifeline for each active participant, where the messages are the operations corresponding to communications, and execution specifications are used to model the private activities of the participants. UML offers also further ways to represent the interactions among the participants, such as the interaction overview diagram (similar to the BPMN choreography), and so the method can be extended with additional ways to represent the interaction views.

As **U1**, **U3** covers both Large Processes and Long Processes.

• **U4** is the lighter version of **U3**, defined as **U3** but now, as for **U2**, the definitions of the various operations are not part of the models, and the participant classes have no attributes.

The model of Buying produced following **U4** is obtained by deleting the methods defining the operations in the one reported in Fig. 7 and 8.

• **U5** extends **U1** to support the enterprise related features. The whole behaviour of the active participants is modelled by means of state machines. The organization units are represented by classes without operations stereotyped by ≪organization≫. The architecture of an enterprise is modelled by a composite structure diagram containing the organization unit classes and the active and business object participant classes (an organization unit class containing some participants or other organization units is rendered by a structured class having as subclasses the classes of the contained entities).

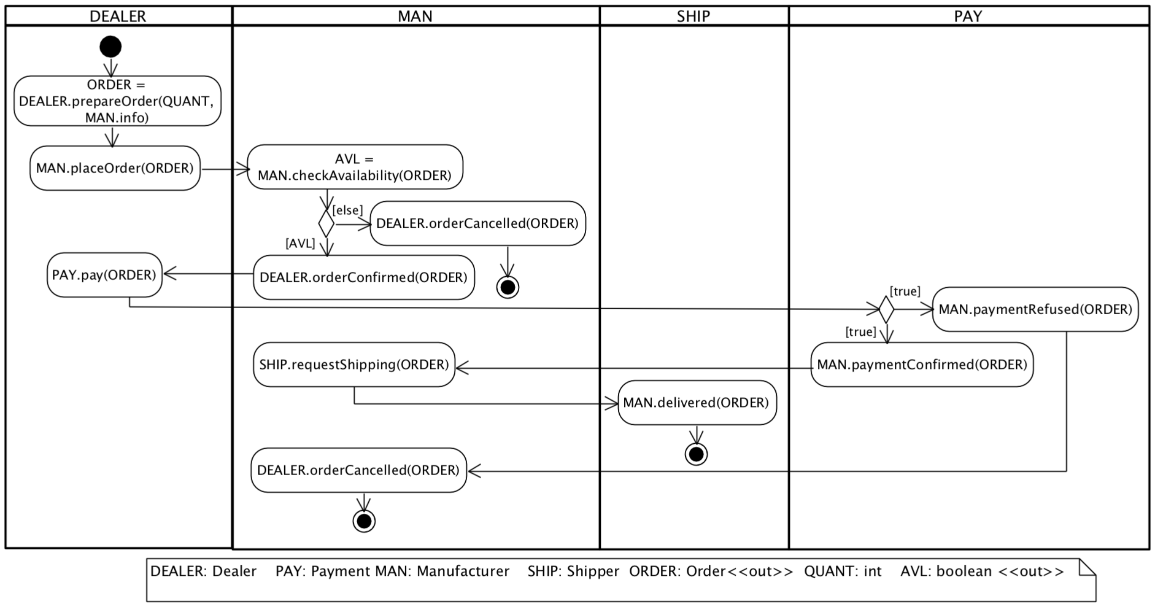

DEALER
MAN
SHIP
PAY
ORDER = DEALER.prepareOrder(QUANT, MAN.info)
MAN.placeOrder(ORDER)
AVL = MAN.checkAvailability(ORDER)
[else]
DEALER.orderCancelled(ORDER)
[AVL]
PAY.pay(ORDER)
DEALER.orderConfirmed(ORDER)
[true]
MAN.paymentRefused(ORDER)
[true]
SHIP.requestShipping(ORDER)
MAN.paymentConfirmed(ORDER)
MAN.delivered(ORDER)
DEALER.orderCancelled(ORDER)
DEALER: Dealer PAY: Payment MAN: Manufacturer SHIP: Shipper ORDER: Order<<out>> QUANT: int AVL: boolean <<out>>

Fig. 8. **U3** model of the Buying process: Workflow View

The process overview is modelled by a use case diagram, where use cases stereotyped by ≪process≫ represent the processes, and the actors represent the process participants.

Finally, the goals (both of the enterprise and of the processes) are modelled by use cases stereotyped by ≪goal≫, their mutually relationships (e.g. dependency) by associations among them, and a stereotyped association will link the processes with their goals.

The model of Buying and of the enterprise enacting it is too large to be reported in this paper, it can be found in [14].

D. Comparing BPMN and UML on modelling cases

In this initial proposal we try to compare BPMN and UML on five modelling cases: ① Data-aware processes machine processable; ② Data-aware processes human readers; ③ Executable processes; ④ User-manual; and ⑤ Processes of an enterprise. Table II reports a column for each modelling case showing the required features, while two adjacent columns show which features are supported by the best BPMN and UML based methods for such case, respectively. The best methods have been selected as described in the following subsections.

① *Data-aware processes machine processable.* Nowadays data and business objects are gaining a first-class citizen status in business process modelling (see e.g. [22], [23]), moreover, data-aware process models may be the starting point to develop big-data/analytics projects to improve them. Machine-processable means that the produced models should be processable by software tools, for example to transform them into code or inputs for simulation software, obviously without human intervention (this means that all the relevant information must be provided by the model), thus the models should be at least Hard. Data-aware processes require the features reported in Table II.

B1 (but also **B2**) only partially supports the selected features (no Business Object Structure & Behaviour, Data Definition and Task Behaviour, and partially Task Participants), and it does not seem possible to define another BPMN-based method overcoming these limitations (see the discussion at the end of the definition of **B1**); whereas **U1** supports all the features.

② *Data-aware processes human readers.* Similar to ①, but now the models are intended for human readers, so they may be loose, and participants and tasks do not need to be detailed. Summarizing this case requires the features reported in Table II.

B1 and **U2** essentially support all the features listed above; also **U1** is able to support the selected features, but it will result unnecessarily heavy for what concerns the modeller effort.

③ *Executable processes.* If the model of a business process is intended to be the starting point for developing a software system to manage the process itself or to provide the input to a simulation software, the tasks should be “executable” and their behaviour should be modelled; moreover, it is important to distinguish between human and system participants, and all aspects of the business objects and of the data participants should be modelled. This modelling case requires the features reported in Table II.

U3 fully supports this modelling case, whereas **B2** only partly supports it; however, the lacking features are not realizable using the BPMN, thus we cannot work out another method supporting all the features required in this case.

④ *User-manual.* The intended way for the users to interact with a software system is a business process, and its model realized with a visual notation may be an effective way to communicate it, i.e. it provides an easy to understand user manual. The tasks are executable, the details about the participants and the tasks are not relevant, both the workflow and the interaction views are required, and the models may be loose. The features required by the user-manual business process are reported in Table II. **B2** and **U4** support all the required features.

⑤ *Processes of an enterprise.* This is the case where all the processes of an enterprise should be modelled, and so to

TABLE II
SUITABILITY OF BPMN AND UML FOR FIVE MODELLING CASES

	Modelling cases														
	Data-aware process machine processable			**Data-aware process for human readers**			**Executable process**			**User-manual**			**Enterprise's processes**		
Features	***Case***	**U1**	**B1**	***Case***	**U2**	**B1**	***Case***	**U3**	**B2**	***Case***	**U4**	**B2**	***Case***	**U5**	**BPMN?**
Participants (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Active participants (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Workers & Systems (E)							✓	Yes	P				✓	Yes	
Active Participant Structure & Behaviour (V)													✓	Yes	
Business Objects (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Business Object Structure & Behaviour (V)	✓	Yes	☒				✓	Yes	☒				✓	Yes	
Data (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Data Definition (V)	✓	Yes	☒				✓	Yes	☒				✓	Yes	
Organization Units (E)													✓	Yes	
Tasks (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Task Participants (E)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Task Behaviour (V)	✓	Yes	☒				✓	Yes	☒				✓	Yes	
Abstract tasks (E)	✓	Yes	P	✓	Yes	P									
Executable tasks (E)							✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Workfow View (V)	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	Yes	✓	Yes	
Interaction View (V)							✓	Yes	Yes	✓	Yes	Yes			
Messages (E)							✓	Yes	Yes	✓	Yes	Yes			
Goals (E)													✓	Yes	☒
Multiple Processes (E)													✓	Yes	
Process Overview (V)													✓	Yes	☒
Entity View (V)													✓	Yes	☒
Enterprise Architecture (E)													✓	Yes	
Long Processes (D)				✓	Yes	P									
Large Processes (D)				✓	Yes	P									
Formal (C)															
Hard (C)	✓	Yes	☒				✓	Yes					✓	Yes	
Soft (C)															

✓ required empty: NOT required **P** partially supported **Yes** supported

cannot be covered by any extension of the method: ☒

Feature kind: **E** (Element) **D** (Dimension) **V** (View) **C** (Characteristic)

guarantee their mutual consistency all aspects of that enterprise must be modelled. All the features required by this case are reported in Table II.

U5 is essentially a modelling method for the enterprises and so it supports all the required features, whereas it is not possible to devise a BPMN based method supporting the enterprise specific features, as stated explicitly in the official specification [1, sect. 7.1]: BPMN is only for modelling the business processes.

• ***Results of the comparison.*** The result of the comparison, see Table II, is that UML covers more modelling cases than BPMN; in our opinion it is motivated by the following facts:

1) UML is a general purpose notation with diagrams and constructs able to cover many different aspects, as the definition of data, objects and active participants, whereas BPMN has been designed with a restricted specialized scope;
2) UML is easily extendible using the profile mechanism, indeed, all the proposed UML based methods introduced in this paper use specific UML profiles defined by sets of stereotypes, e.g. to model the goals;
3) UML provides also languages for expressing the textual inscriptions (OCL and the UML actions with concrete syntax) with a defined semantics and means to define data and objects, whereas BPMN, for example, by definition does not include a part to express the data;
4) BPMN is in some sense incomplete since some fundamental parts of the models may be only provided using some tools dedicated to produce and execute the BPMN models (e.g. the `InputOutputSpecification`s of the tasks), and using non-standard languages, so there is a big difference in the expressive capabilities of BPMN as a pure notation and of BPMN supported by a tool. Thus, comparing UML with a BPMN (non-standard) variant supported by a tool we may obtain different results.

Summarizing, our method-wise approach that allows to select the most suitable method and thus a notation for a specific modelling case, can be very useful for the modellers.

V. RELATED WORKS

In the literature there are many papers trying to compare different modelling notations for the business processes, for example [5]–[7], [10], [11], [24]–[26], but only [12] shares our point of view that modelling notations cannot be compared in isolation, indeed it aims to help to select the most appropriate for the specific case "based on the purpose and type of model". In general, the interesting results in the cited papers and in many others are extremely useful to people concerned with the design of modelling notations, but they offer a scarce support to modellers having to face specific modelling cases.

Table III summarizes the business process model elements and views proposed in the cited references and their correspondence with our features.

The literature concerning business process modelling methods based on BPMN is almost non existing. One, and to the best of our knowledge the sole, proposal is reported in [20]. The Hagenberg business process modelling method [27] supports, as us, a holistic view of business processes that cannot modelled using only the BPMN; indeed [27] proposes additional and integrated notations, for example for data and actors.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we described an approach aimed at selecting the most suitable business process modelling notation for a given modelling case. The main novelty is in the perspective of the approach. Indeed our proposal, differently from many others focusing either on specific properties of the notations or on their constructs, further raises the level of comparison. The idea is to compare "modelling methods" that are based on a notation rather than "modelling notations". Indeed, when a modeller has to face a modelling case, they will not look for the best notation per se, but for the best method based on a notation they know and according to a set of characteristics specific of the modelling case to handle. Thus, on the one hand, our work proposes and describes a set of features that can be often associated with common modelling cases. On the other hand, it describes a set of modelling methods based on the BPMN and UML notations that support a subset of the features. At this point the modeller has to just select which are the features of interest for their specific modelling case, and then to find a modelling method that supports them.

The features considered in this work are based on our experience in modelling business processes in the academic and industrial domains [14], and in our research activities on this topic, but they subsume the "features" proposed in the literature (see Table III).

TABLE III
BUSINESS PROCESS MODEL ELEMENTS AND VIEWS IN THE CITED REFERENCES

Feature	**Aldin et al.** [9]	**List et al.** [10]	**Adamo et al.** [24]	**Hagenberg** [27]
Workflow view	process			process model
Task	activity	activity	set of activities	
Participants	role	process participant[1]		
Business objects versus data			information vs carrier	data model
Active participants			agentive vs non agentive participants	
Organizational units			organizational boundaries	organizational
Participants $\neq$ business entities			object vs role	
Goals	goal	goal	goal/value	
Subsumed by goal	service & product	service & product		
Subsumed by task	events	events		
Subsumed by task behaviour	rules			
Interaction view				dialog model
Entity view				actor model
covered by object/system/date participants		resource		
in/out process participants			process clear input/output	

We plan to perform a field validation. More in detail, we will propose to the experiment participants several real examples (found on the web or suggested by professionals) of business process to model. Then, the participants will apply our approach. In this way we will evaluate (1) the effectiveness of the approach in producing a satisfactory model and (2) the effort required to apply the approach. The inclusion of both BSc/MSc/PhD Students and Professionals among the participants will provide insights on the effectiveness of the approach when carried out by subjects with different level of expertise. Moreover, at the end of the study: (1) new methods will be developed if the available ones do not allow to handle new cases (notice, that in general the modellers do not need to develop such methods, these tasks should be delegated to modelling experts), (2) new mappings modelling case $\Leftrightarrow$ fit method mappings will be added to the catalogue, and (3) new features needed to characterise the new cases may arise, helping to tune the approach.
Finally, we plan also to consider other modelling notations, e.g., coloured Petri nets that are formal and can be validated using well-established tools, as the CPN-tool; unfortunately, they have a quite limited set of constructs and so they are difficult to use on realistic cases. But, it is may be possible to devise clever methods overcoming these limitations.

REFERENCES

[1] OMG, *Business Process Model and Notation (BPMN) 2.0*, 2011.

[2] ——, *Unified Modeling Language (UML). Version 2.5.1*, 2017.

[3] R. Davis and E. Brabander, *The Event-driven Process Chain*. Springer, 2007, pp. 105–125.

[4] K. Jensen and L. M. Kristensen, *Coloured Petri Nets – Modelling and Validation of Concurrent Systems*. Springer, 2009.

[5] D. Peixoto, A. B. Vitor, A. P. Atayde, E. P. Borges, R. F. Resende, and C. I. P. S. Padua, "A comparison of BPMN and UML 2.0 activity diagrams," in *Proc. of SBQS 2008*, 2008.

[6] C. V. Geambasu, "BPMN vs. UML Activity Diagram for Business Process Modeling," *Journal of Accounting and Management Information Systems*, vol. 11, no. 4, pp. 637–651, 2012.

[7] D. Birkmeier, S. Kloeckner, and S. Overhage, "An empirical comparison of the usability of BPMN and UML activity diagrams for business users," in *Proc of ECIS 2010*. AIS, 2010, p. 51.

[8] J. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska, "Measuring method complexity: UML versus BPMN," in *Proc. of AMCIS 2009*. AIS, 2009.

[9] L. Aldin and S. de Cesare, "A comparative analysis of business process modelling techniques," in *Proc. of UKAIS 2009*. AIS, 2009.

[10] B. List and B. Korherr, "An evaluation of conceptual business process modelling languages," in *Proc of SAC 2006*. ACM, 2006, pp. 1532–1539.

[11] J. L. Pereira and D. Silva, "Business process modeling languages: A comparative framework," in *New Advances in Information Systems and Technologies*. Springer, 2016, pp. 619–628.

[12] R. S. Aguilar-Savén, "Business process modelling: Review and framework," *International Journal of Production Economics*, vol. 90, no. 2, pp. 129–149, 2004.

[13] G. Reggio, E. Astesiano, and C. Choppy, "A framework for defining and comparing modelling methods," in *Software, Services, and Systems*, ser. LNCS, vol. 8950. Springer, 2015, pp. 377–408.

[14] G. Reggio and M. Leotta, "Precise Business Process Modelling @DIBRIS Web Site," 2019, available at `sepl.dibris.unige.it/2019-PBPM.php`.

[15] G. Reggio, M. Leotta, D. Clerissi, and F. Ricca, "Service-oriented domain and business process modelling," in *Proceedings of 32nd ACM/SIGAPP Symposium on Applied Computing (SAC 2017)*. ACM, 2017, pp. 751–758. [Online]. Available: https://doi.org/10.1145/3019612.3019621

[16] G. Reggio, M. Leotta, and F. Ricca, ""Precise is better than Light" A Document Analysis Study about Quality of Business Process Models," in *Proceedings of 1st International Workshop on Empirical Requirements Engineering (EmpiRE 2011)*. IEEE, 2011, pp. 61–68. [Online]. Available: https://doi.org/10.1109/EmpiRE.2011.6046257

[17] E. Astesiano, G. Reggio, and F. Ricca, "Modeling business within a UML-based rigorous software development approach," in *Concurrency, Graphs and Models*, ser. LNCS, P. Degano, R. DeNicola, and J. Meseguer, Eds., vol. 5065. Springer, 2008, pp. 261–277.

[18] G. Reggio, M. Leotta, F. Ricca, and E. Astesiano, "Business Process Modelling: Five Styles and a Method to Choose the Most Suitable One," in *Proceedings of 2nd International Workshop on Experiences and Empirical Studies in Software Modelling (EESSMod 2012)*. ACM, 2012, pp. 8:1–8:6. [Online]. Available: https://doi.org/10.1145/2424563.2424574

[19] G. Reggio, F. Ricca, G. Scanniello, F. D. Cerbo, and G. Dodero, "On the comprehension of workflows modeled with a precise style: results from a family of controlled experiments," *Software and System Modeling*, vol. 14, no. 4, pp. 1481–1504, 2015.

[20] B. Silver, *BPMN Method and Style, 2nd Edition*. Cody-Cassidy, 2011.

[21] OMG, *Object Constraint Language (OCL) Version 2.4*, 2014.

[22] D. Cohn and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes," *IEEE Data Eng. Bull.*, vol. 32, 2009.

[23] A. Marrella, M. Mecella, A. Russo, S. Steinau, K. Andrews, and M. Reichert, "Data in business process models, a preliminary empirical study," in *Proc. of SOCA 2015*, 2015, pp. 116–122.

[24] G. Adamo, S. Borgo, C. D. Francescomarino, C. Ghidini, N. Guarino, and E. M. Sanfilippo, "Business process languages: An ontology-based perspective," in *Proc. of JOWO 2017*, ser. CEUR, vol. 2050. CEUR-WS.org, 2017.

[25] B. Axenath, E. Kindler, and V. Rubin, "An open and formalism independent meta-model for business processes," in *Proc. of BPRMO 2005*, 2005.

[26] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N.Russell, "On the suitability of BPMN for business process modelling," in *Proc. of BPM 2006*. Springer, 2006, pp. 161–176.

[27] F. Kossak, C. Illibauer, V. Geist, C. Natschlge, T. Ziebermayr, B. Freudenthaler, T. Kopetzky, and K.-D. Schewe, *Hagenberg Business Process Modelling Method*. Springer, 2016.