Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Hybrid process modelling languages

Maryam Ilyas

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :
dr. Niels MARTIN

2019
2020

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Hybrid process modelling languages

Maryam Ilyas

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

dr. Niels MARTIN

Towards a comprehensible resource-aware hybrid process modelling language

Maryam Ilyas

Hasselt University, Agoralaan Building D, 3590 Diepenbeek, Belgium
maryam.ilyas@student.uhasselt.be

Abstract. Within a business context, it is essential to communicate a process effectively and efficiently. A process can be visualised by a process model that represents the relations between the activities, executed by non-human or human resources. The literature provides a multitude of process modelling languages that can be divided into two categories (i.e. procedural and declarative languages). These polar categories either focus on the structured relations or unstructured relations within a process. Although the use of an individual modelling language is not sufficient to provide a comprehensible process model. To overcome this limitation, various hybrid languages are proposed. Most of the hybrid languages only focus on the control perspective of a process and neglect the resource-related relations. Within this context, the paper outlines the development of a resource-aware hybrid process modelling language. The research follows the principles of the Design Science approach to build and evaluate the artefact that solves a general problem. An empirical study is conducted to evaluate the artefact. The designed artefact, BPMND+R, is based on an existing hybrid modelling language which is extended to support the resource perspective of a process. BPMND+R is evaluated on its comprehensibility compared to BPMN and Declare, in a theoretical and empirical study.

Keywords: Hybrid process modelling language · Resource perspective · Flexible processes.

1 Introduction

Rapid technological advancements bring changes to the activities executed within a company. As the execution of each activity has an impact on another, it is necessary to understand the interaction between the different activities. Once the relations between the activities are understood, the company can identify the various processes that are performed within the company [63]. These processes are referred to as business processes in the literature. A business process is defined as "a collection of inter-related events, activities and decision points that involve several actors and objects, which collectively lead to an outcome that is of value to at least one customer" [21].

Business process modelling (BPM) is an approach to display how organisations execute their business processes [32]. Business process models visualise the relations between the activities that are performed by human or non-human resources within a business context [33,76]. These models can be designed with different purposes. For instance, several domain experts use an interactive model to analyse 'what-if' questions related to the attributes of the process. These models focus on the dynamic and functional aspects of the process [5]. BPM can also be used to assist people in understanding, describing, specifying and documenting a process more effectively than they can do using natural language [35]. Lastly, a process model can be used to analyse the process by assessing its properties which leads to process re-engineering and improvement [51]. The focus of this research is on the process models built to communicate a process.

A process modelling language provides a set of semantics to systematically represent a business process [51]. The literature introduces a multitude of process modelling languages. These can be divided into two categories: procedural and declarative languages. Procedural languages represent a 'close world' as the process model captures all possible activity flows. Hence any unspecified activity flow is disallowed [14]. Activities can also have a more flexible relationship; this means that there is not always a defined sequence in which the activities should be executed. This relates to an 'open world' assumption, as everything is allowed unless it is explicitly forbidden by a constraint [14]. Modelling languages representing this behaviour are referred to as declarative process modelling languages. In practice, a process is never fully structured or unstructured. While some activities within a process might have more structured relations, there might be other activities for which the relations cannot be defined using only procedural notations [59]. A hybrid approach could provide a reasonable solution for such processes. The declarative paradigm could be used for the flexible parts of the process and the procedural paradigm for specifying strict relations. A hybrid process modelling language combines existing languages at the level of their syntax, semantics and language paradigm [6].

Hybrid languages offer a balance between the declarative and procedural paradigm [6]. Besides, the process modellers and domain experts are more familiar with the flow-based notations such as BPMN, which makes constraint-based declarative languages less desirable for these users. A hybrid approach will allow them to use these declarative languages in combination with the control-flow based models, resulting in more comprehensible models [46,81]. However, hybrid languages are not fully developed yet. On the one hand, the hybrid languages merely focus on the control-flow aspect of a process and ignore the relations between the activities and their resources. On the other hand, the developed hybrid languages are not empirically tested regarding their comprehensibility compared to procedural and declarative languages.

This paper introduces BPMND+R, an extension of an existing hybrid language. This research extends the control-flow perspective of the hybrid language and provides semantics to support the resource perspective of a process while retaining the comprehensibility of the hybrid language. The comprehensibility of the proposed extension is evaluated in a theoretical and empirical study, compared to BPMN and Declare.

The remainder of this paper is structured as follows. The next section (Section 2) defines the research questions and methodology of the research. Then, Section 3 gives an overview of the related work. Subsequently, in Section 4, several requirements for the artefact are constructed based on the literature. Section 5 introduces an extension of a hybrid process modelling language that also visualises the resource perspective of a process. Finally, this solution is evaluated in Section 6, regarding its comprehensibility for novice users compared to BPMN and Declare.

2 Research questions and methodology

2.1 Research questions

The main question which triggers this research is: How can existing hybrid process modelling languages be extended to visually communicate the control-flow and resource perspective of a process in a comprehensive way for human users? This general research question can be divided into multiple sub-questions.

What are the different process modelling languages? The research looks into different process modelling languages. These can be divided into two categories: procedural and declarative languages. These two categories are also the building blocks of a hybrid language. Therefore, it is essential to understand the advantages, disadvantages and the differences between the two paradigms. Accordingly, a few principal procedural and declarative languages are explored.

What are the various hybrid process modelling languages? The literature examines the different hybrid process modelling techniques. The focus hereby is, to explore languages that combine procedural and declarative notations to model a process. The research also evaluates the different hybrid languages on their ability to model the control-flow and resource perspective of a process.

How can BPMN-D be extended to visualise the resource perspective of a process? The focus of the research is to extend a hybrid process modelling language to visualise the resource perspective of a process. Consequently, it is crucial to consider how exactly the resource perspective can be visualised. These visualisations could either be inspired from an existing process modelling language or based on an independent language, built to visualise the resource-related relations within a process.

How comprehensible is BPMND+R? The study looks into the literature to understand how a process modelling language can be evaluated on its comprehensibility. Subsequently, the introduced extension is evaluated regarding its comprehension. On the one hand, different theoretical techniques are sought. On the other hand, the focus is on an empirical approach.

2.2 Methodology

This research will follow the Design Science approach [29]. The Design Science methodology can be defined as: "Design science is the scientific study and creation of artefacts as they are developed and used by people to solve practical problems of general interest" [29]. The focus of Design Science is to develop artefacts to overcome research problems. The outcome of a Design Science research is not only the artefact and the applied method but also the contextual knowledge that explains the design principals, why the method works and under which conditions. Therefore, the two main activities of this methodology are: building and evaluating the artefact. Following the Method Framework introduced in [34], a Design Science study undergoes six phases. These phases are shortly discussed in the next section in regard to this research [34].

1. At the beginning of a Design Science research, a problem is explicated. This problem should be valid for a global practice and not only for the local practices. Within this research, the problem is formulated by reviewing the existing literature related to hybrid modelling languages. The literature study is conducted as follows:

   (a) Most of the literature is based on scientific sources (e.g. books, scientific journals). The keywords used to look up the literature can be divided into three groups: 1) terms referring to procedural, declarative and hybrid languages 2) terms referring to resource patterns within a process and their visualisations 3) terms referring to business process model evaluations, guidelines and empirical study. Both backward and forward-searching was used to include a wide rang of literature [11].

   (b) The inclusion criteria can be defined as follows: 1) the study introduces process models built for communication of the process 2) the study uses graphical notations to model a process. Research is excluded if the following criteria apply: 1) the study is not in English 2) the study uses a technical approach to model the process and evaluate the model [11].

   (c) Next, the abstract and conclusion of the paper were read critically to determine the usability of study [11].

   (d) After a quick scan through the data collected, the concepts will be categorised in a table which enables the comparison of different research papers containing the same concepts. This matrix enables the constant comparison of the data within the same categories [82].

   As the different hybrid languages are already developed, they still miss the ability to support the resource perspective of a process, preserving their comprehensibility. This problem does not only point out the lack of academic research regarding the resource perspective of the hybrid languages, but it is also a missed opportunity for the professional modellers [29,59].

2. Secondly, the artefact in development is defined by specifying its requirements. These requirements can be seen as the demands that the developed

artefact should fulfil to be qualified as a solution to the defined problem [29]. The literature research leads to two types of requirements for the artefact: 1) additional control-flow patterns 2) elementary resource-related patterns.

3. The artefact is then designed and developed according to the specified requirements [29]. Notations of a hybrid process modelling language are extended conform to the defined requirements. Consequently, the proposed artefact includes additional control-flow constructs and resource semantics to model a hybrid process.
4. Next, the artefact is demonstrated. The demonstration of the artefact requires the use of the developed artefact in an illustrative or real-life case. This shows that the artefact can solve the defined problem [29]. For the demonstration of the artefact developed in this research, a semi-structured process is modelled using the artefact. This hybrid model shows that the developed artefact can be used to model a hybrid process and also represent its resource-related relations.
5. Lastly, the evaluation of the artefact measures whether it complies with the predefined requirements [29]. This study evaluates the artefact on its ability to represent the relations between the activities and their resources within a process to its users, in a comprehensive way. This is executed by comparing a hybrid model to a procedural and a declarative model. This evaluation is carried out in two-fold. First, the models are assessed by a theoretical evaluation. Then, the models are examined in an experimental study. This experimental approach is used to investigate causal relations between two or more variables. The variables investigated in this paper are the comprehensibility and the process model. The experimental research is conducted using a questionnaire that includes open and close questions. Consequently, both qualitative and quantitative data is collected and analysed using the appropriate methods (e.g. descriptive or statistical statics, discussion). Nevertheless, internal and external validity should be taken into consideration when examining the results. The empirical study is elaborately discussed further in the research [29].

3 Related work

First, this section gives an overview of the different procedural and declarative languages. Then, it focuses on the different hybrid process modelling languages. Lastly, this section provides an overview of the several empirical studies related to the comprehensibility of a process model.

3.1 Procedural and Declarative process modelling languages

Most of the process modelling languages can be divided into two categories, procedural and declarative. The procedural languages are used in a structured context, such as an ordering process, where the rules between the different activities are prescribed, and every situation is known at the design time [60]. Conversely, declarative languages are more suitable for flexible processes such as a process in health care or education [44].

Consider two activities (A and B) that can occur multiple times in a process, but they exclude each other. After the first occurrence of activity A, activity B cannot occur in the process anymore (same goes for B). This behaviour can be easily modelled using declarative modelling languages, as they solely focus on the relations between the activities. Procedural languages would require more specifications and assumptions to model the described behaviour. For example, the decision of which activity would be executed A, B or none of these, and how many times the selected activity will be executed [77]. These specifications tend to over-specify the behaviour, by precisely describing *how* the process is executed, while declarative languages focus on *what* should be done to achieve the business goals [73].

If any change occurs within the process, the procedural model would have to be re-designed to incorporate the change. Conversely, the declarative process models would only require adding the constraints that represent the change. Another notable difference between procedural and declarative modelling is: how a given behaviour can be classified as satisfying to the model. In a procedural model, a specific behaviour can be reconstructed from the description by finding a continuous path within the model that exactly represents the behaviour. In a declarative model, it is sufficient that the given behaviour satisfies all the constraints of the model. There is no correspondence required between the behaviour and the model [22].

The following sections provide a concise overview of the common procedural and declarative modelling languages.

3.1.1 Procedural process modelling languages

The literature is flooded with different procedural modelling languages such as Event-Process-Chains (EPC) [1], Yet Another Workflow Language (YAWL) [74], Petri nets [52], Unified Modelling Language (UML) [9] and Business Process Modelling Notations (BPMN) [81]. EPC provides general notation rules, different functions and a set of views on single parts of a company [1]. YAWL is a combination of workflow patterns and an extension of the Petri nets formalism [74]. Petri Nets is a mathematical modelling language used for modelling business processes and workflows as place/transition nets. Transitions represent actions, and places signify the different states within the model [52]. UML is for specifying, visualising, constructing and documenting the artefacts of software systems. Nonetheless, UML activity diagrams are frequently used for process modelling [9]. EPC, YAWL and BPMN provide a wide range of constructs to model different perspectives of a process [66].

BPMN is an ISO standard for modelling business process and the de-facto standard for representing process models in professional practice. It is very expressive and has been used in every kind of organisation to express their models, such as the Nobel Prize assignment process, incident management, and e-mail voting systems. BPMN is formal and easily understandable by the domain experts and final users [12]. It provides a wide range of notations, such as activities, events, control-flows, gateways and lanes [81].

Figure 3.1 provides an example of a BPMN model. The process is modelled in an expanded swimlane (represented by a large rectangle including the activities) which represents the resource of the process. This process is executed entirely by Department A with the input of an external actor, which is modelled using a collapsed swimlane. The external actor directly influences the process, but the organisation cannot control its behaviour. The start and end of the process are modelled using start and end events. The model includes different constructs, such as activities, events, and gateways. First, the activity a leads to a parallel-split gateway, where activity b and c are executed simultaneously. Activity c is a collapsed sub-process (represented by activity with a plus-sign on the bottom) which includes multiple tasks. These activities are eventually merged by a parallel-merge. Next, a message is sent using a throwing intermediate message event to an external stakeholder of the process. The main difference between activities and events is that activities are actions executed by the resources, while events are occurrences resulting from activities within the process. A replay is received from the external actor, modelled using a catching intermediate message event. Then, an XOR-split gateway ends the process, or leads to activity e before ending the process. BPMN also supports the data-related relations within a process. It provides constructs, such as a data file and database, which can be attached to an activity (activity a) using a dotted line.

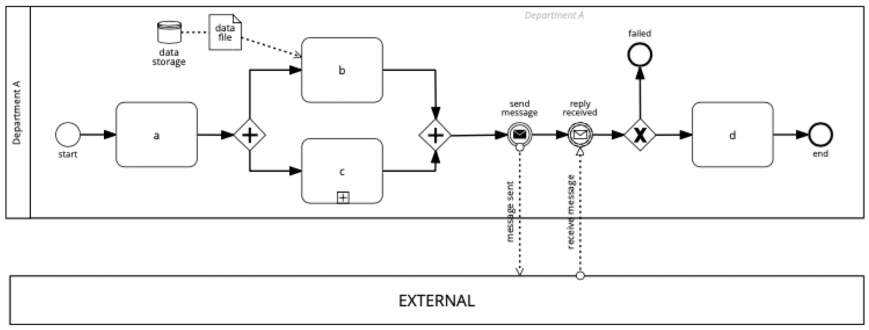


Fig. 3.1: An example of BPMN model

3.1.2 Declarative process modelling languages

Declarative languages are used to model flexible processes [27]. Most of the declarative languages are either constraint- or rule-based. A constraint-based language focuses on individual relations between the activities, and models merely the constraints of a process. Declare [55], Dynamic Condition Response graphs (DCR graphs) [30], Guard-Stage-Milestone (GSM) [31], and Case Management Model and Notation (CMMN) [45] are examples of constraint-based declarative languages. The rule-based languages are text-oriented as they describe business rules in natural language [38]. These languages do incorporate the different perspectives of a process, but they do not provide any graphical notations to model the process. Therefore, these are not included in this study. This section provides an overview of several constraint-based declarative languages.

PENELOPE consists of constraints based on permissions and obligations. The term *Perm*(Buyer, PlaceOrder) represents the permission for a buyer to place an order [26].

DeciClare is a mixed-perspective declarative language, because it also includes constraints regarding the data, resource and time perspective of a process. The DeciClare constraints are based on Declare constraints. The authors evaluate the Declare constraints to their defined requirements and extend these Declare constraints when necessary [50].

Business Process Constraint Network (BPCN) does not only provide a set of sixteen constraints to model the unstructured process, but these can also be modelled using graphical notations [43].

ADEPT workflow system allows its end-user to modify a process model during the execution (i.e. add, delete and change the sequence of tasks) while preserving several control-flow and data-flow consistencies [57].

Worklets are often used to handle specific tasks in an extensive process. They have a binary tree structure, where the nods represent if-then statements (if condition, then conclusion). These if-statements are evaluated ripple down and eventually execute tasks that are described within the then-statement [4]. The worklets are maintained within an extensible repository, such that at runtime a preferred worklet is contextually chosen to fulfil the activity goal.

Declare focuses on the individual relations between the activities rather than the whole flow of the process. It makes sure that the execution of the process is compliant with the functional specifications of the process by specifically defining the relations between the activities represented by constraints. Declare provides a large set of constraint templates, divided into three groups: existence, relation and negation constraints [54,55]. The Declare constraints are usually expressed in Linear Temporal Logic [10,25]. LTL uses temporal operators such as next time ($\circ$ F), eventually ($\diamond$ F), always ($\Box$ F), and until (F $\sqcup$ G) to describe the relations between the activities. For example, the LTL expression (A $\rightarrow \diamond$ B), specifies that any execution of activity A is eventually followed by activity B.

These semantics are hard to understand for a beginner who is not familiar with LTL semantics. Consequently, there are different approaches to visualise the Declare constraints, such as DecSerFlow [77]. DecSerFlow notations can be used to model workflows of dynamic processes. Table 3.1 provides several Declare notations with their templates and semantics. These notations provide different type of arcs to model the behaviour between two activities (A and B).

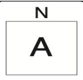
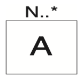
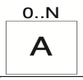
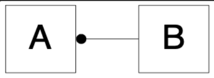
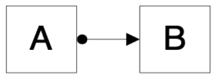
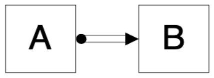
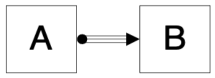
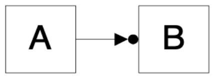
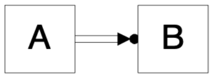
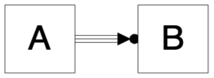
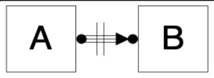
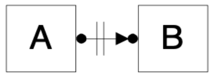
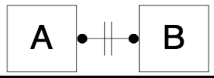
Notation	Template	Semantics
	Exactly_N(A)	A must occur exactly N times
	Existence_N(A)	A must occur at least N times
	Absence_N(A)	A can occur at most N times
	Responded existence(A, B)	If A occurs, B must occur as well
	Response (A, B)	If A occurs, B must eventually follow
	Alternate response(A, B)	If A occurs, B must eventually follow, without any other occurrence of A in between
	Chain response(A, B)	If A occurs, B must occur next
	Precedence(A, B)	B can occur only if A has occurred before
	Alternate precedence(A, B)	B can occur only if A has occurred before, without any other occurrence of B in between
	Chain precedence(A, B)	B can only occur immediately after A
	Not chain succession(A, B)	if A occurs, B cannot occur next
	Not succession(A, B)	if A occurs, B cannot eventually follow
	Not co-existence(A, B)	if A occurs, B cannot occur

Table 3.1: Declare notations, semantics and templates

Figure 3.2 gives an example of a Declare model, visualised by DecSerFlow notations given in Table 3.1. This model consists of five activities and four relations between these activities. Activity a and activity b, are connected with an alternate precedence constraint. This constraint implicates that every time b is executed, it has to be preceded by the execution of activity a. Co-existence constraint specifies that if b occurs in the process, d has to be executed as well (before or after b). The other way around, if d occurs in the process, b has to be executed as well. Furthermore, the execution of activity d would require an immediate occurrence of activity e. Lastly, activity c is eventually executed after an occurrence of activity a.

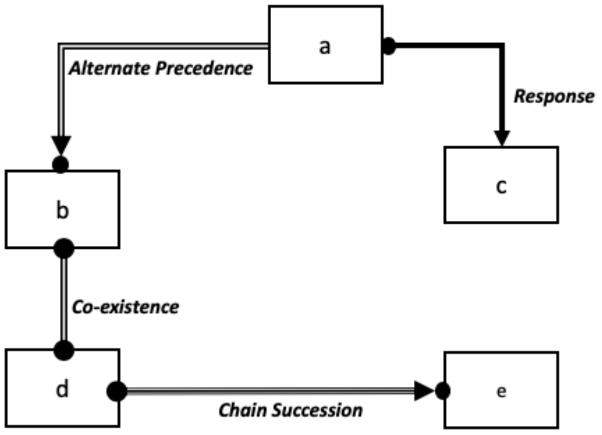


Fig. 3.2: An example of Declare model

The Declare models can contain hidden dependencies between its constraints. Hidden dependencies are the interaction between constraints and their activities without being explicitly modelled in the process design [15]. For example, in Figure 3.2, activity d has a hidden dependency with activity a and activity c. If activity d occurs, activity b has to be executed as well because of the co-existence constraint between activity b and d. But activity b has to be preceded by activity a (alternate precedence constraint), consequently if activity a is occurred, eventually activity c has to be executed as well (response constraint). This shows a hidden dependency between activity d and c, without being explicitly modelled in the model. Several papers introduce solutions to recognise these hidden dependencies. However, the solutions are merely based on mathematical algorithms and do not explain the graphical notations [15,83].

Dynamic Condition Response graphs (DCR graph) is an emerging process modelling language in the declarative paradigm. It contains four types of binary behaviours, which makes it comprehensible relative to Declare. These binary behaviours are represented by four relations which are response, condition, milestone and exclude/include relation [30]. These relations are briefly described below and illustrated in Figure 3.3.

- Response relation between A and B (depicted graphically as $\cdot \rightarrow$) states that if A happens, B eventually has to happen to complete the workflow. This brings the flexibility that B does not have to happen for every execution of A. For example, in Figure 3.3, if event *a* occurs, event *b* has to be executed eventually.
- Condition relation between A and B (depicted graphically as $\rightarrow \cdot$) states that before B can happen, A must have happened before. For example, event *d* can only happen after the occurrence of event *b* (Figure 3.3).
- Exclude relation between A and B (depicted graphically as $\rightarrow \%$) allows dynamically excluding events from the workflow. This represents an XOR-split, as the two events mutually exclude each other. For example, in Figure 3.3, event *d* excludes itself, consequently *d* cannot be executed more than once in the process. Include relation between A and B (depicted graphically as $\rightarrow +$) represents that B cannot happen until an occurrence of A. This way, the occurrence of A includes B into the model. Figure 3.3 shows a dashed event (event *c*), that is not included within the process at the beginning. If event *b* is executed, the event *c* will automatically be included in the process.
- Milestone relation (depicted graphically as $\rightarrow \diamond$) is used in correspondence with nested event. Nested event is a logical grouping of activities, such as a sub-process. In Figure 3.3, the large rectangle X represents a nested event, including event *b*, *c* and *d*. Only if the execution of all the events inside this nested event *X* is completed, then event *e* can be executed.

DCR graphs represent the control-flow and resource perspective of a process. It visually assigns a resource to an activity. This can be seen in Figure 3.3, where the rectangles right above the activities provide the resource of each activity.

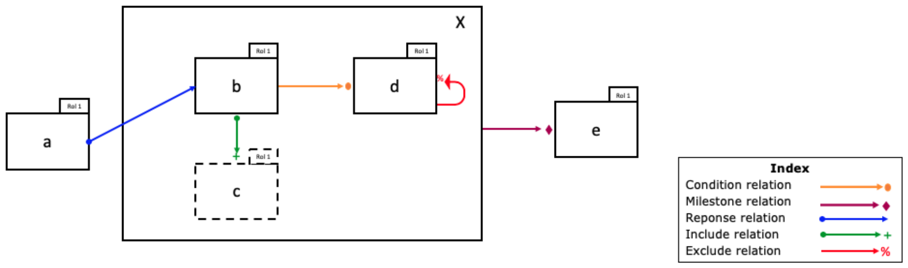


Fig. 3.3: An example of DCR graph

Guard Stage Milestone (GSM) notations consist of stages, milestones and guards. A stage represents an atomic task. It can have one or more guards, which are the conditions that need to be satisfied before a stage can be activated. Also, one or more milestones can be linked to a stage. While a guard controls when and how a stage may start, the milestone defines when and how a stage may end [31].

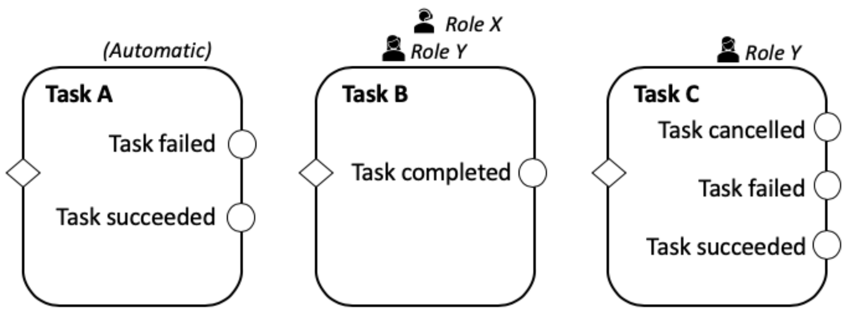


Fig. 3.4: An example of GSM model

Figure 3.4 represents an example of a GSM model. This example contains three stages (represented by rectangles). A guard (⋄) is attached at the left side of the stage. In this example, the stages can start when the preceding stage has reached one of its milestones. Milestones (∘) are attached to the right side of the stages. These represent the different acceptance criterion of the stages. Thus, *Task A* can fail or succeed, and this will mark the end of this task. GSM also provides the resources of a task (above each stage). GSM notations are used to provide the life cycle of a process. These semantics are elementary and cannot be used to model a complicated process [31].

Case Management Model and Notation (CMMN) provides declarative notations based on GSM semantics. Models built with CMMN are referred to as cases, such a model is presented in Figure 3.5. The process is modelled using tasks, stages, milestones and event listeners. These are the work items of a case, presented in a collapsed case file. An entry criterion (⋄) describes the condition that must be satisfied for a work item to be available for execution. An entry criterion (⋄) attached to the border of the case file initiates the process in Figure 3.5. A milestone represents the accomplishments of a case instance. The milestone *received* in Figure 3.5, for instance, marks the entry criterion of the case file as received and start the collapsed stage. Stages are used as sub-processes, which can be executed depending on the type of instance. The stage in Figure

3.5 includes two tasks. A task in a case can either be non-blocking or blocking. A non-blocking task is considered completed when the resource takes up the task. Blocking task, however, should be explicitly marked as completed by the resource. In this example, the stage has two non-blocking tasks. In short, the tasks within the collapsed stage are immediately executed at the start of the case file. CMMN also provides notations for the data perspective of a process. In Figure 3.5, for example, a data file represents the entry criteria of the blocking task *c*, which is required to reach the *Completed* milestone. Lastly, the timer event is used as the exit criterion (⧫) of the case file in Figure 3.5. An exit criterion is the opposite of an entry criterion, as it specifies when to stop working on the stage, task or case [45].

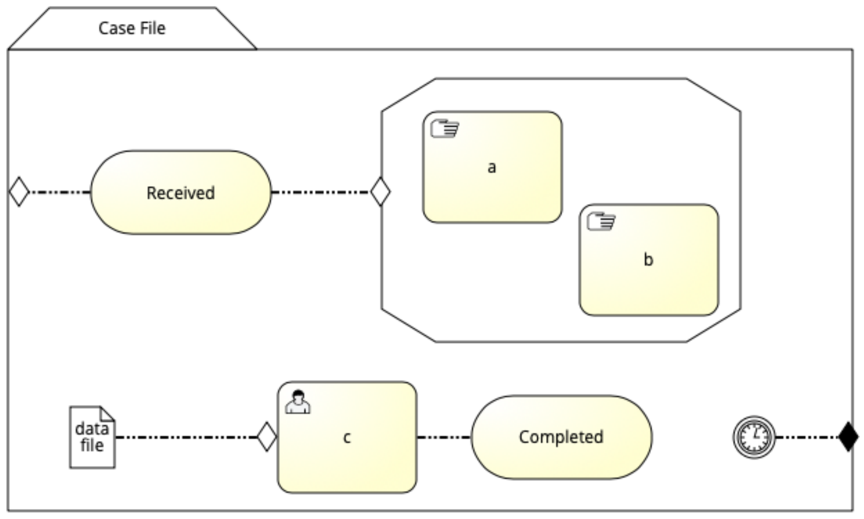


Fig. 3.5: An example of CMMN model

3.2 Hybrid process modelling languages

Hybrid languages provide flexible, adaptable and comprehensible models for complex processes. A hybrid approach gives a reasonable solution for a process which includes structured and unstructured relations between the activities. The declarative languages can be used to model flexible relations and procedural languages for structural control-flows.

Abbad Andaloussi et al. (2020) provide an extensive literature review on the different hybrid approaches. The authors analyse 30 hybrid process modelling languages and recognise two types of hybrid approaches in the literature: hierarchical or fully mixed approach. A hybrid model based on a hierarchical approach consists of a core- and a sub-process, each of them modelled with either a procedural or a declarative language. To the contrary, within the fully mixed approach, the notations of procedural and declarative languages fully overlap in the same model. Some of the authors use mixed structures to design their hybrid language . The study mentions a large group of hybrid languages that consists of procedural models combined with rule-based declarative languages [6]. As mentioned earlier, these languages are not included in this research.

3.2.1 Pockets of Flexibility

Pockets of Flexibility is a hierarchical hybrid language. The hybrid process model consists of a predefined structured workflow and pockets of flexibility. A pocket of flexibility includes several activities which have variable relations. The flow of these activities is only specified during the run time of the process. Pockets of flexibility does not only ensure flexibility but also process maintainability, as the process model does not have to change entirely or remodelled to incorporate future changes. Figure 3.6 gives an example of a workflow with a pocket of flexibility. The pocket of flexibility is designed using a special workflow activity, build activity. It consists of several fragments (x,y,z). The number of executions of these fragments and their relations are only specified during the run time of the process. The execution of these fragments within the build activity is completely dependent on the process instance. An instance template is a particular composition of the fragments defined at the run time [62]. Figure 3.7 shows two instance templates of the process model in Figure 3.6. These instance templates can be used for a specific type of instance which requires the same behaviour as modelled by the instance template [62].

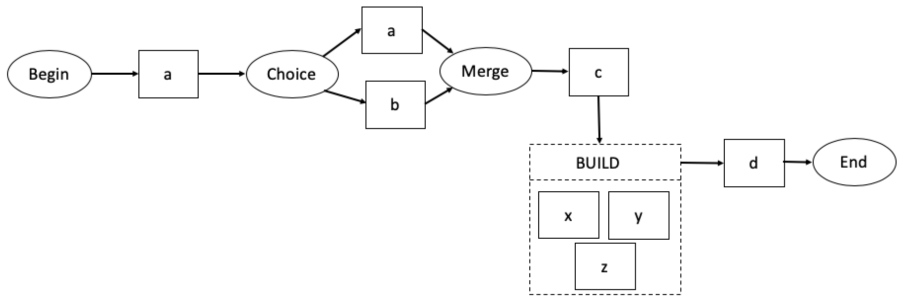


Fig. 3.6: An example of Pockets of Flexibility model

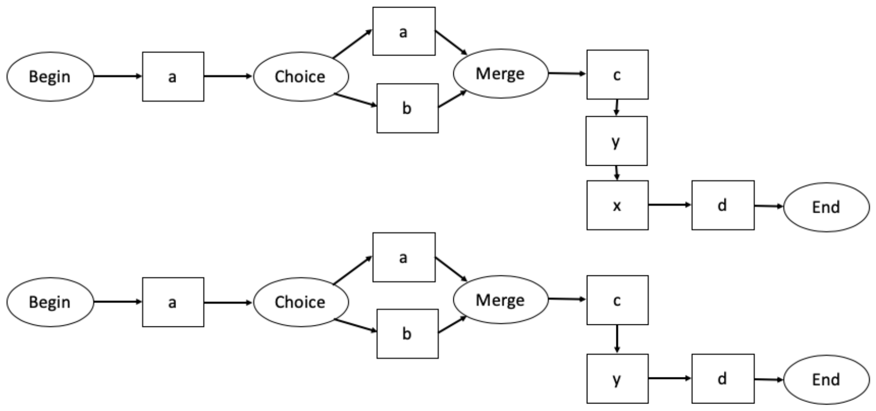


Fig. 3.7: Instance templates

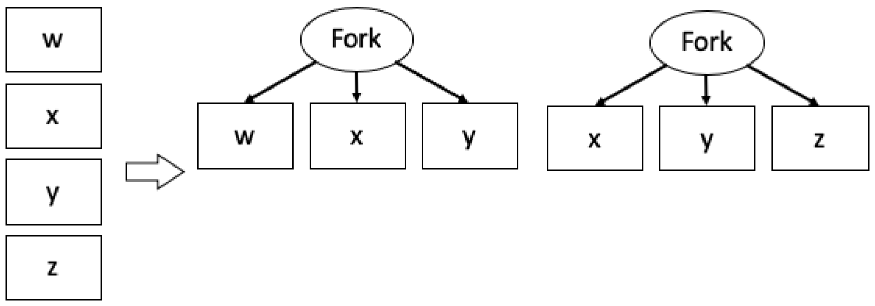


Fig. 3.8: Building fork constructs

The fragments within the build activity can have different compositions such as sequential or sorted as a fork. The sequential composition is used in the instance templates given in Figure 3.7. These fragments are sequentially executed during the run time. Conversely, the fork construct in Figure 3.8 can be compared to the AND-split construct of BPMN. The instance templates containing a fork structure needs to synchronise (AND-join) the different flows, using immediate synchronisation (Figure 3.9.a) or deferred synchronisation (Figure 3.9.b). Immediate synchronisation represents that all the activities within the fork have to synchronise before the instance can continue with the control-flow. Deferred synchronisation gives the flexibility that the instance does not have to wait for all of the activities within the fork to proceed further. Eventually, all multiple branches will merge before the core-process completes. Lastly, the fragments in build activity can also be executed in arbitrary or multiple iterations. Arbitrary executions are based on do-while or repeat-until constructs. Multiple executions specify that a fragment is executed sequentially or in a fork construct multiple times [6].

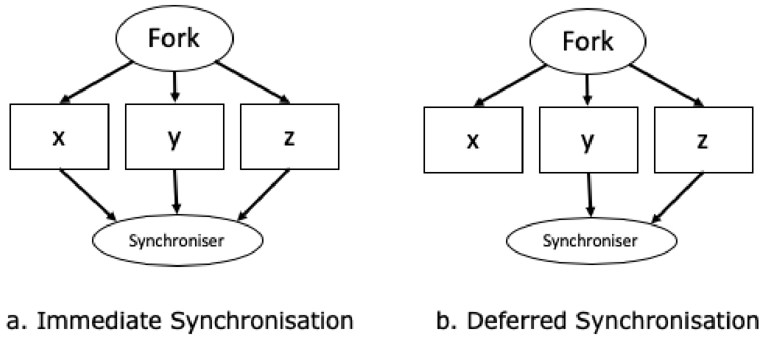


Fig. 3.9: Building Synchronisation constructs

3.2.2 Hybrid process trees

Another hierarchical approach is hybrid process trees where the top-level of the process is modelled using process trees and the flexible sub-processes are presented by DCR graphs. Figure 3.10 provides an example of the hybrid model. This hybrid process tree consists of four types of control flow (sequential, exclusive, parallel and redo) and three types of activities (abstract, silent and atomic). The abstract activities (*X1, X2*) represent flexible sub-processes that are modelled using DCR graphs, inducing flexibility into the model [30,58].

3.2.3 Hybrid process modelling language with Declare

Declare models can get complex when it comes to a structured process. A large amount of constraints decreases the comprehensibility of the model [15]. Nevertheless, Declare is suitable for the flexible parts of a process. Different studies use Declare to model the less-structured parts of the process in combination with a procedural language.

YAWL, Worklets and Declare

For example, van der Aalst et al. (2009) combine Declare with YAWL. The hierarchical hybrid language consists of a procedural core-process, modelled by YAWL [74] and multiple sub-processes which are either modelled by Declare [55] or Worklets [4]. By combining the three languages, this hybrid approach supports

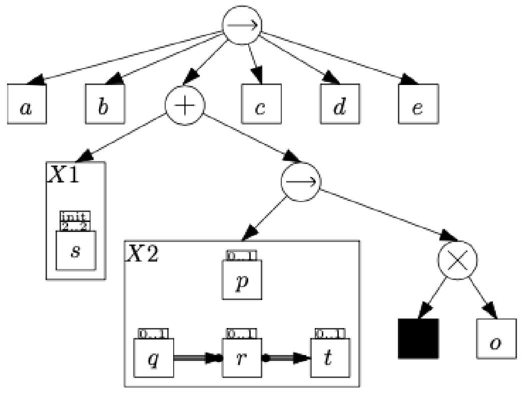


Fig. 3.10: An example of Hybrid model with Process tree and DCR graph

variations of flexibility. Figure 3.11 gives an example of a hybrid language. The core-process is modelled using YAWL notations. YAWL specifies the control-flow and prevents the need for change and deviation. Activities in the core-process activate the sub-processes which are modelled using Worklets or Declare. Worklets allow for flexibility by underspecification as it provides an extensible repertoire of actions at run time. The sub-process which merely consists of rules is modelled using Worklets (Figure. 3.11 (b)). Declare is used for sub-processes which has a few execution constraints and many possible execution paths. These sub-processes can easily be altered by including or removing a constraint. Declare provide flexibility by design, deviation, and change [2].

Petri nets and Declare – Hierarchical approach

Several studies combine Petri nets and Declare to design their hybrid languages [16,17,72]. Slaats et al. (2016) propose a hierarchical approach, where the core-process is designed using Petri nets and sub-process can either be modelled using Petri nets or Declare, depending on their characteristics. Figure 3.12 represents a hybrid model of an order-fulfilment process. In this model, the label on the activity consists of two parts: the actions and the actual label. The actions represent the executable elements of the process, while the labels represent the names of these actions. The label *quality check* is used twice within this process, for action *c* and action *e*. At action *c*, the label *quality check* triggers a sub-process which is modelled using Declare. Action *e* with the same label (*quality check*) leads to another task. The difference between activities and labels is necessary to ensure independence between the actions which might have the same label. This includes flexibility within the process as the labels can be used multiple times within the process model without creating any confusions [72].

Petri nets and Declare – Fully mixed approach

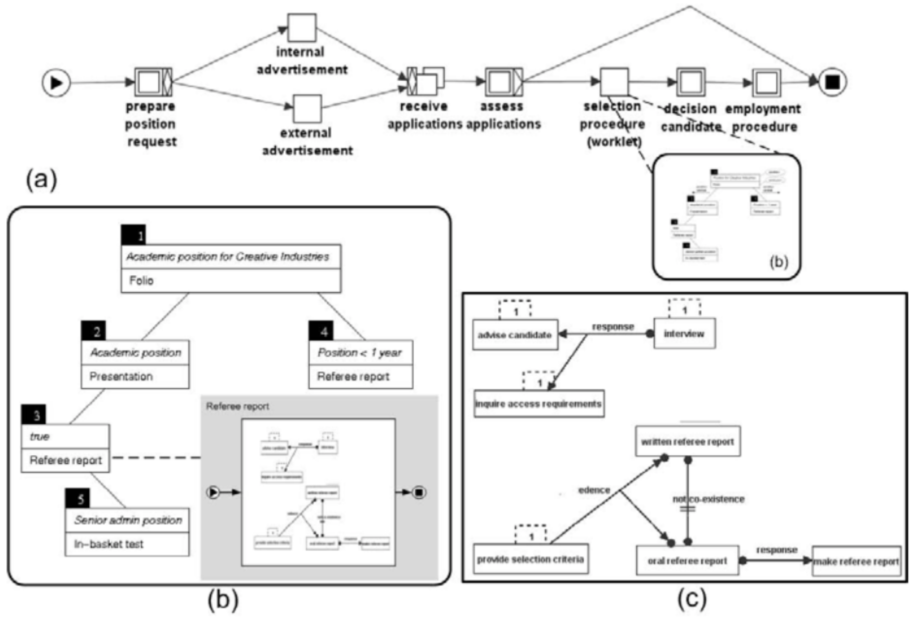


Fig. 3.11: An example of hybrid model with (a) YAWL, (b) Worklets and (c) Declare model

Petri nets and Declare are also combined with a fully mixed approach. Figure 3.13 presents a hybrid model that contains structured relations, modelled using Petri nets (blue arcs) and unstructured relations modelled using Declare constraints (black arcs). For example, the relation between activity A is always followed by activity B, and activity B is always preceded by activity A. The relation between activity A and B can easily be modelled using sequential flows of Petri nets (blue arc between A and B). Other activities might have an unstructured relation, such as an activity D that can only occur if Z has occurred before, without any other occurrence of activity D in between (alternate precedence) [77]. The relation between activity D and Z is complex and modelled using a Declare notation. The combination of Petri nets and Declare notations allows the hybrid model to also contain Existence constraints. For instance, the number *1* on the start place of the model represents *Exactly(1)* constraint. This implicates that start activity can only be executed once in the process [16,17].

Coloured Petri nets, Declare and DCR graphs

Another fully mixed hybrid approach is introduced by combining Coloured Petri nets (CPN) with Declare and DCR graphs. CPN is specially designed to add the data perspective within the usual Petri nets [8]. This hybrid language is one of the few which focuses on the data and resource aspects of the process. Although the model was introduced in the context of process simulation and the semantics

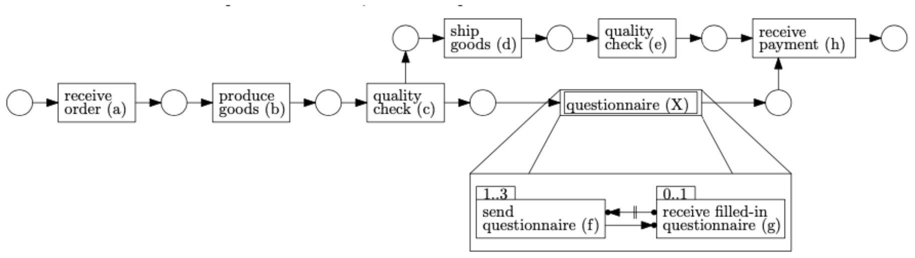


Fig. 3.12: An example of hybrid model with Petri nets and Declare

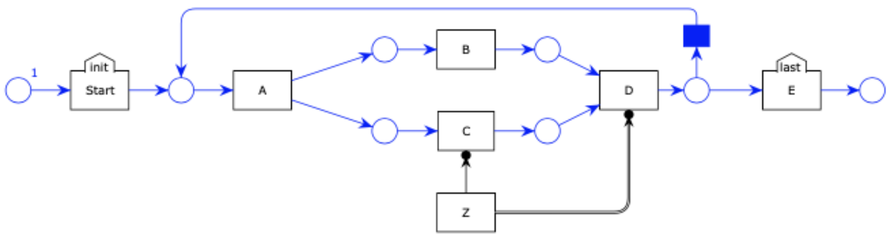


Fig. 3.13: An example of Hybrid model with Petri nets and Declare

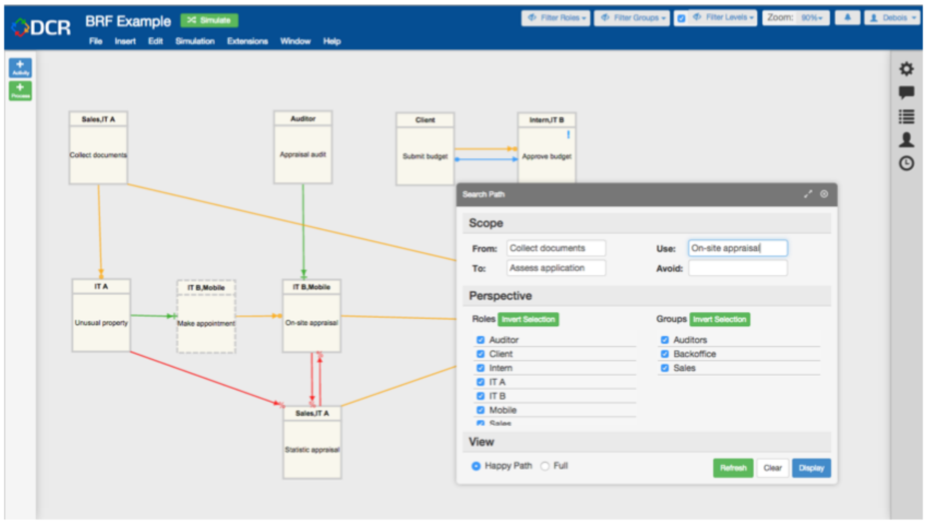
of the process are not elaborated [80]. Thus, the notations of both the control-flow and resource perspective are not discussed within the paper.

3.2.4 BPMN and DCR graphs

DCR graphs and BPMN are merged in a hierarchical structure where the DCR graph represents the core-process. The DCR graph represents a large number of admissible traces within the process. The user can zoom into one specific path by using a search function. Within the search function, the user specifies which activity would mark the start and the end of the concerned single-path. As a result, this search function presents a BPMN model which represents the flow between the specified start and end activities. The search function in Figure 3.14 includes the flow between the activity *Collect documents* and *Assess application*. Consequently, Figure 3.15 provides the BPMN model resulted from the search function specifications. With the provided search function and 'the happy-path' in BPMN, the user can focus on the trace of their concern [18].

3.2.5 BPMN-D

BPMN-D [14] is a fully mixed approach based on BPMN and Declare. The basic concepts of BPMN, such as activities and control-flows, are extended with a declarative approach. BPMN-D also uses XOR-gateways from BPMN to route the different flows within the model. This hybrid language introduces four types



Scope

From: Collect documents
To: Assess application
Use: On-site appraisal
Avoid:

Perspective

Roles
- [✓] Auditor
- [✓] Client
- [✓] Intern
- [✓] IT A
- [✓] IT B
- [✓] Mobile
- [✓] Sales

Groups
- [✓] Auditors
- [✓] Backoffice
- [✓] Sales

View

[✓ Happy Path] Full

Fig. 3.14: An example of DCR graph(left) and the search function(right)

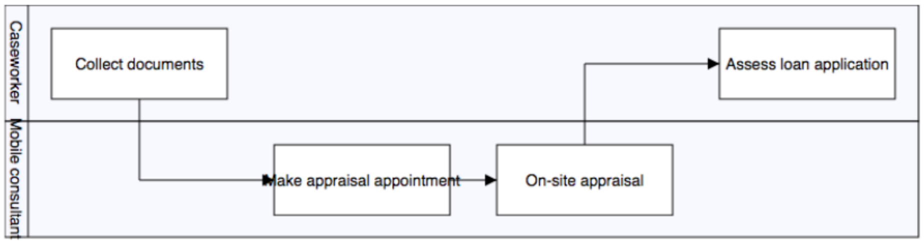


Fig. 3.15: The resulting single trace in BPMN

of tasks and four types of arcs to design a process model. The limited number of constructs makes BPMN-D easy to understand for a novice.

The following four types of activities (represented in Figure 3.2 (left)) are introduced:

1. Atomic task is an individual task executed during the process. This task is equivalent to the atomic task in the BPMN notations.

2. Inclusive task includes a range of tasks given in a list. The user can choose which task will be performed during the run time. It represents the XOR-choice from BPMN. The user can execute only one task from the given list.

3. Exclusive task provides a range of tasks that cannot be executed during the run time. The user can perform any task apart from the tasks listed.

4. Any task represents that the user can perform any task available within the process context.

The following four types of arcs (represented in Figure 3.2 (right)) are introduced:

1. Sequence flow is the same as the control-flow arc of the BPMN notations. The activities linked to this flow are executed sequentially.
2. Inclusive flow represents a collection of tasks. The user can choose one or more tasks from the given list and perform it with zero or more repetitions before proceeding to the next modelled task. This represents the OR-choice from BPMN because the user can simultaneously execute one or more tasks.
3. Exclusive flow provides a collection of tasks. The user can choose one or more tasks apart from the given list and perform it with zero or more repetitions before proceeding to the next modelled task.
4. Any flow gives the user freedom to perform any task in the process context, with zero or more repetitions, before proceeding to the next modelled task.

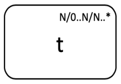
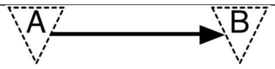
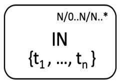
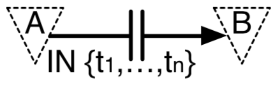
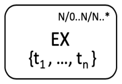
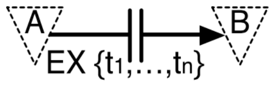
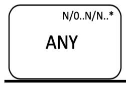
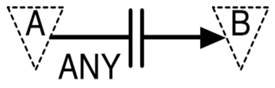
Notation	Name	Notation	Name
	Atomic task		Sequence flow
	Inclusive task		Inclusive flow
	Exclusive task		Exclusive flow
	Any task		Any flow

Table 3.2: BPMN-D notations: tasks (left) and control-flows (right)

3.3 Evaluation of the model

Prior research has shown that the visual representation of a process model can have a significant impact on the cognitive load of a human user, without changing the information that is provided within the modelled process [37]. There is a vast amount of literature which investigates the factors that influence the comprehension of the business process models. This literature can be categorised into empirical and theoretical work for procedural languages.

3.3.1 Empirical work

The empirical studies (e.g. experiments, questionnaire studies) are often executed to determine factors that influence the comprehensibility of a process model and to compare two or more modelling languages. These empirical studies define process models as the research object and comprehension as a dependent variable. The indicators for measuring comprehension of a model can either be objective or subjective [23]. There are two types of objective indicators to measure the comprehensibility of a process model: 1) Objective comprehension accuracy 2) Time taken to understand the model. The objective comprehension accuracy is measured using comprehension questions such as multiple-choice questions with correct/incorrect answers or problem-solving based on the model content. The efficiency is measured by recording the time taken to understand a model. The subjective indicators are measured using questionnaire scales like perceived ease of understanding, perceived usefulness, and preference ratings. The results from these studies are analysed with a variety of statistical methods such as ANOVA test, Kruskal-Wallis test, Spearman's correlation and Pearson's correlation analysis. Moreover, students are the participants of most of the empirical studies. Nevertheless, there is input from the domain experts and the process model experts from academia or practice, but it is limited. [23]

Process modelling languages have been widely evaluated and compared with regard to their understandability. Table 3.3 provides a summary of several empirical studies concerning different procedural and declarative languages. As there are different studies that evaluate the procedural languages, there are only a few that evaluates the comprehensibility of the different declarative languages. An empirical study that was intended to compare Declare and DCR graphs discovered the need for the hybrid process models. The participants urged that the use of declarative notations might be more attractive if these were used in combination with procedural languages.

When it comes to hybrid approaches, there is not much empirical work executed yet. Most of the existing empirical studies are limited to the understandability of textual hybrid process artefacts [42,78,83]. These languages do not provide graphical notations for the declarative part of the hybrid model. Therefore they are not included within this study.

3.3.2 Theoretical work

There are different modelling guidelines and quality frameworks concerning model comprehensibility. For example, Guidelines of Modelling (GoM) includes six principals: correctness, clarity, relevance, comparability, economic efficiency, and systematic design. For instance, to apply the relevance guideline, the model designer should eliminate model elements as long as the model does not lose its meaning [7]. Consequently, this guideline is difficult to measure objectively for a novice. SEQUAL is a quality framework to assess how well a process achieves its goals [75]. Also, this framework is hard to measure and use for a novice user. For

Evaluated process modelling language(s)	Research subjects	Conclusion	
BPMN	Professional modellers	The study recommends five guidelines to enhance the comprehensibility of the BPMN models.	[40]
Declare	Students, postdocs, and professors	The authors conclude that the combination of several Declare constraints make the Declare models complex for its users.	[28]
Declare	Students	The study concludes that the end-users can effectively use Declare models, but which only includes a considerable spectrum of constraints.	[79]
DCR graphs	Researchers and practitioners	The majority experienced the DCR graph notations easy to understand.	[46]
UML, BPMN, Petri nets, YAWL	Practitioners	The study discovered that none of the language top-ranked in comprehensibility and perceived acceptance. Though the evaluation by expert users identified BPMN as the preferred notation.	[66]
EPC and Petri nets	Practitioners	The authors conclude that EPC elements are better understood than Petri nets.	[67]
BPMN and Declare	Students	The study concludes that procedural process models are more comprehensible than declarative process models.	[56]
Declare and DCR graphs	Practitioners	The authors conclude that both Declare and DCR graphs notations are too academic and neither convincing nor intuitive for practitioners. The authors also discovered the need for the hybrid process models.	[59]

Table 3.3: Overview of empirical studies conducted to measure the comprehensibility of procedural and declarative languages

example, this framework requires correctness as a precondition of executability. For procedural models, soundness is a measure for correctness. Therefore, to employ this guideline, the user requires knowledge on soundness and how it can be measured for different modelling languages. 7PMG is a set of seven process modelling guidelines. These guidelines provide recommendations on how to build a process model from scratch as well as for improving existing process models. Each of these guidelines is built on reliable empirical insights. Consequently, these guidelines are easy to follow for a non-expert process modeller [47]. These guidelines are shortly discussed further.

G1: Use as few elements in the model as possible. The size of a model strongly influences the comprehensibility of the process [47]. If a process model has more than 65 nodes (activities, gateways and events), it is considered highly inefficient and complex to understand. If the number of nodes is between 31 and 37, the model is considered efficient and comprehensible for its users [64]. Mendling et al. (2012) confirm this threshold by also suggesting not to use more than 31 nodes within a process model [48].

G2: Minimise the routing paths per element. The higher the number of input and output arcs, the harder it becomes to understand the model [47]. Mendling et al. (2012) recommend not to model more than three inputs or outputs per connector [48]. Whereas, Sánchez-Gonzálezet al. (2012) concludes that each decision node should have fewer than 7 to 9 input or output sequence flows [65].

G3: Use one start and one end event. The number of starts and end event is positively correlated with the complexity of a process model [47]. Mendling et al. (2012) provide a threshold of 1 start and 1 end event [48].

G4: Model as structured as possible. Structured models can be seen as formulas with balanced brackets, where each opening bracket has a corresponding closing bracket of the same type. A process model is structured if every split connector matches a respective join connector of the same type. Thus, gateway mismatches, where the split gateways type does not match the join gateway type, has a negative correlation with the understandability of a process model [47,64]. However, Dumas et al. (2012) conclude that structuring might be beneficial only when it does not increase the number of gateways [20].

G5: Avoid OR routing elements. An OR routing element bring ambiguity within the process model and influence its comprehensibility [47]. The study by Sarshar and Loos (2005) also concludes that tasks related to OR routing symbols are more complicated than those related to AND and XOR, without giving the exact numbers [67]. Sánchez-González et al. (2012) suggest including no more than 10 XOR, 7 AND and 4 OR decisions. It provides a low number for OR decision nodes because the OR decision nodes make a process model more complex than the XOR and AND decision nodes [65].

G6: Use verb-object activity labels. A verb-object style (e.g. "Analyse report") is significantly less ambiguous and more useful than action-noun labels (e.g. "Report analysis") or the labels that follow neither of these styles [47]. Also, the design of a label influences model comprehension such as size, direction, colour and position of the text. Nevertheless, it is not empirically evaluated yet [39]. Furthermore, Mendling and Strembeck (2008) report that the longer the labels, the lower the comprehension accuracy, as they increase the effort required to read these labels [49].

G7: Decompose the model if it has more than 50 elements. Larger models increase the complexity of the model. Therefore, they should be split up into smaller models. Tasks with a single entry and a single exit can be replaced by one activity which points to a separate model including the original tasks [47,64]. Mendling et al. (2012) specifically suggest decomposing a model with more than 31 nodes [48].

These guidelines give a good overview of the quality aspects for the understandability of the process models. Nevertheless, these criteria are based on several empirical studies which included only procedural process models. Thus, their applicability remains questionable for declarative process models [6].

4 Defining requirements

A process modelling language tends to support one or more perspective of a process. Firstly, the following sections describe the control-flow and resource perspective of a process shortly. Next, the procedural, declarative and hybrid process modelling languages are evaluated on their ability to provide semantics for the resource and the control-flow perspectives of a process. Lastly, these evaluations are used to define the requirements of the artefact in development.

4.1 Control-flow perspective

Declare provides a spectrum of constraints that support different relations between two activities. The following sections define the control-flow perspective of a process. Then, the hybrid languages are evaluated on their ability to support the control-flow perspective of a process by comparing them to the Declare constraints.

4.1.1 Defining control-flow perspective

Declare constraints can be divided into three groups: existence, relation and negation constraints (represented in Table 3.1). These groups of constraints can be briefly described as following:

- Existence constraints specify the number of executions for an activity. For instance, Exactly_N(a) specifies that activity a must be executed exactly

N times within the process. Existence_N(a) constraint implicates that activity a can be executed minimum N times within the process. Conversely, absence_N(a) specifies that activity a can be executed maximum N times within the process.

– Relation constraints define relations between two activities. For example, alternate precedence constraint specifies that *activity b* can only occur if *activity a* has occurred before without any other execution of *activity b* in between.

– Negation constraints specify the negative relations between two activities. Not co-existence constraints, for instance, represents the XOR-choice. Thus, it specifies that if a occurs, b cannot occur in the process.

4.1.2 Evaluating the control-flow perspective of hybrid languages

The control-flow behaviour of the introduced hybrid languages is evaluated by comparing them to Declare constraints given in Table 3.1. Table 4.1 gives an overview of the different hybrid languages and whether they support any template from a specific group of Declare constraints. For example, BPMN-D supports several relations (e.g. alternate precedence, response) and negation constraints (not chain succession, not co-existence) but it does not support any existence constraints.

It is noticeable that if a hybrid language does not directly include Declare notations, it does not support any existence constraints. The function of the existence constraints is two-fold as on the one hand, these templates restrict the behaviour of a model. On the other hand, they add flexibility within a model because an activity can be skipped, once it has reached the minimum number of executions. Therefore, these constraints coincide with the hybrid behaviour employed in this research.

Hybrid languages	**Existence constraints**	**Relation constraints**	**Negation constraints**
Pockets of flexibility		✓	✓
Hybrid process trees		✓	✓
YAWL Worklets, Declare	✓	✓	✓
Petri nets and Declare - hierarchical	✓	✓	✓
Petri nets and Declare - fully mixed	✓	✓	✓
BPMN and DCR graphs		✓	✓
BPMN-D		✓	✓

Table 4.1: Control-flow support of Hybrid process modelling languages compared to Declare constraints

4.2 Resource perspective

This section specifies the resource perspective of a process. Then, the previously discussed procedural, declarative and hybrid languages are evaluated on their ability to support the described resource patterns.

4.2.1 Defining resource perspective

The resource perspective represents relations between the activities and human or non-human resources [21]. Russell et al. (2005) and Mertens et al. (2017) define different resource patterns, such as direct allocation, role-based allocation, organisation allocation, supervision, two-role allocation, separation/binding of duties and organisation structure [50,61]. These patterns can be described as following:

- Direct allocation determines the resource that will execute a task.
- Role-based allocation is the ability to specify that a task can only be performed by resources that correspond to a given role.
- Organisation allocation assigns an activity to a resource based on their organisational position.
- The execution of activity requires supervision from another resource.
- The two-role allocation directly supports the 4-eye principle. The activity is executed by two resources working together.
- The separation of duties specifies that two activities cannot be executed by the same person. While binding of duties represents that if an activity is performed by a resource, the other activity should also be performed by the same resource.
- The organisation structure gives an overview of the different roles from various departments, assigned to resources.

4.2.2 Evaluating the resource perspective of existing languages

Different procedural languages provide graphical notations for the resource perspective of a process. Conversely, many declarative languages allow their users to define the resources at the design time but they do not provide any graphical notations for this resource pattern [68]. In Table 4.2 the discussed declarative languages are evaluated on their ability to define the various resource patterns. For example, Declare and CMMN support the direct and role-based allocation but these allocations are not presented in the process models [45,55]. Variations on Declare, such as DeciClare provides only constraints to model the resource perspective [68]. DCR graphs and GSM do contain the visualisation for resource allocation [30,31].

Hybrid modelling languages are mostly focused on the control-flow perspective and do not discuss other perspectives of a process. Although hybrid language

Resource pattern	Declare	DeciClare	DCR	GSM	CMMN
Direct allocation	✓	✓	✓	✓	✓
Role-based allocation	✓	✓	✓	✓	✓
Organisational allocation					
Supervision allocation					
Two-role allocation				✓	
Separation/Binding of duties		✓			
Organisation structure					
Graphical notations			✓	✓	

Table 4.2: Resource pattern support by declarative process modelling languages

introduced by Debois et al. (2015) does mention the allocation of resources. The introduced hierarchical hybrid approach combines DCR graphs and BPMN. Both of these languages provide graphical notations for resource patterns. Consequently, these resource-related notations are also incorporated in the hybrid model [18].

To sum up, both declarative and hybrid languages lack the visualisations for the resource aspect of a process. Nevertheless, these languages can easily be extended by using independent graphical notations which solely focus on the visualisation of the resource perspective such as RALph. RALph language is highly expressive and independent from other modelling languages [69]. The RALph semantics are specifically designed to support the resource perspective of a process. These semantics directly support the direct allocation, the role allocation, the organisation allocation and the separation/binding of duties. The full overview of RALph semantics is given in Appendix A.

4.3 Artefact requirements

The aim of this paper is to enrich a hybrid process modelling language by extending its control-flow notations and including the visualisations of elementary resource patterns. The artefact in development will include the existence constraints and the described resource patterns. Consequently, Table 4.3 gives an overview of the requirements for the artefact in development. In the following sections, the artefact is designed based on these requirements and eventually validated.

5 BPMND+R

BPMND+R is an extension of an existing hybrid language, BPMN-D which is based on BPMN and Declare. As these two languages are well-known in their respective paradigms, BPMN-D is a viable hybrid language example that represents both paradigms. Moreover, it only consists of 8 constructs, which makes it desirable for beginners. BPMND+R includes additional control-flow semantics

Artefact Requirements
1. The artefact supports the Existence constraints of Declare.
2. The artefact supports the Direct allocation of the resources in the process.
3. The artefact supports the Role allocation of the resources in the process.
4. The artefact supports the Organisation allocation of the resources in the process.
5. The artefact supports the Supervision allocation of the resources in the process.
6. The artefact supports the Two-role allocation of the resources in the process.
7. The artefact supports the Separation/Binding of duties of the resources in the process.
8. The artefact provides an overview of the Organisational structure.

Table 4.3: Artefact Requirements

and resource patterns. Consequently, BPMND+R notations can be divided into two groups: control-flow patterns and resource patterns. These are introduced in the next sections.

5.1 Control-flow patterns

The control-flow patterns of BPMND+R represent activities and control-flows. The activity notations of BPMN-D are extended by including the existence constraints notations of Declare. The control-flow notations are adapted from BPMN-D without alternations.

Table 5.1 represents the activities of BPMND+R with extended notations. The *N/ 0..N/ N..** notations in the right corner are based on the DecSerFlow notations of Declare constraints (represented in Table 3.1). These activities are explained below:

– An atomic task (represented in Table 5.1.a) must be executed exactly (*N*) N times, or it can be executed at most (*0..N*)/at least (*N..**) N times. For example, an atomic task with notation *N..** in the right corner, emphasise that the task should be performed at least N times during the whole process. Once the minimum number of executions is reached, the activity can be skipped.

– An inclusive task (represented in Table 5.1.b) represents XOR-choice. The user executes one task from the given list of tasks, exactly, at most or at least N times.

– An exclusive task (represented in Table 5.1.c) also provides a list of tasks. The user can perform any task apart from the tasks given in the list, exactly, at most or at least N times.

– An any task (represented in Table 5.1.d) emphasises that the user can perform any task within the context of the process exactly, at most or at least N times.

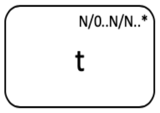
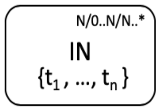
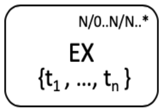
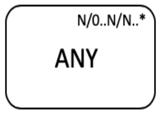
Notation	Description	Notation	Description
 (a)	Perform task t exactly /at most/ at least N times.	 (b)	Perform a task among t_1, . . . , t_n exactly /at most/ at least N times.
 (c)	Perform a task different from t_1, . . . , t_n exactly/ at most/ at least N times.	 (d)	Perform any task exactly/ at most/ at least N times.

Table 5.1: BPMND+R task notations

The described activities can be connected using four types of flows. The flow representations are adapted from BPMN-D and explained in Section 3.2.5. Table 5.2 gives a concise summary of these flows.

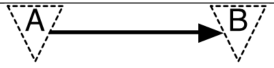
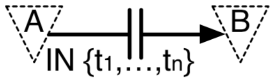
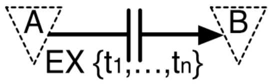
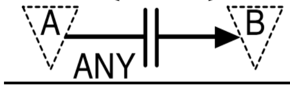
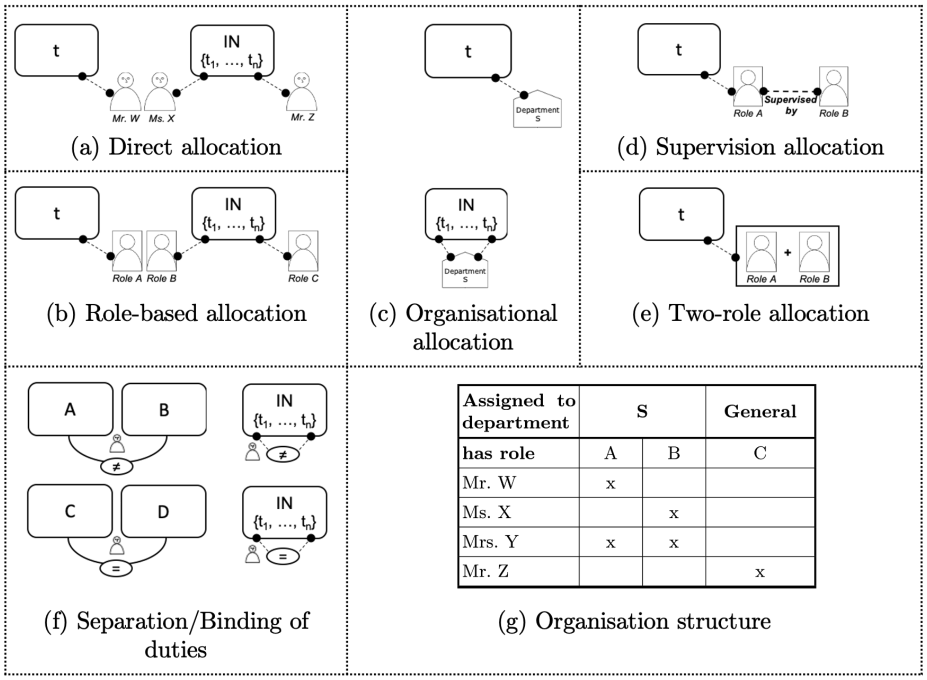
Notation	Description
	B is followed after A.
	B is followed after A, with 0 or more repetitions of tasks t_1, . . . , t_n in between.
	B is followed after A, with 0 or more repetitions of tasks different from t_1, . . . , t_n in between.
	B is followed after A, with 0 or more repetitions of any tasks within the process context in between.

Table 5.2: BPMND+R control-flow notations

5.2 Resource patterns

Figure 5.1 presents the resource-related notations of BPMND+R. These notations can be attached to an atomic task, an inclusive task or an inclusive control-flow. These control-flow patterns represent activities that can be executed within the process, linking resource patterns to these control-flow patterns will specify the relations between the resources and activities executed within the process. The exclusive activity and control-flow, for example, only represent the tasks that should not be executed within the process. Consequently, specifying the resource relations of these tasks would not add any value to the model, as these tasks are not meant to be executed. The any task and control-flow represent all of the tasks within the process context. Each task in a process has a unique relation to its resource, therefore linking resource patterns to any task or control-flow is not practical.



Assigned to department	**S**		**General**
has role	A	B	C
Mr. W	x		
Ms. X		x	
Mrs. Y	x	x	
Mr. Z			x

Fig. 5.1: BPMND+R resource patterns

The resource patterns given in 5.1 can be explained as follows:

– Direct allocation defines which resource will execute a task. In Figure 5.1.a, *task t* is executed by *Mr. W*. The resources can also be attached to specific tasks within an inclusive activity. For instance, the resource *Ms.X* is attached to *task* t_1 and resource *Mr.Z* is attached to *task* t_n (Figure 5.1.a). Hence, if a task among these is executed, the corresponding resource will perform the task.

– The role allocation is based on a role of the resource. These roles can be attached to an atomic task or an inclusive activity. The corresponding role will eventually execute the task within the process. In Figure 5.1.b, *Role A* will execute *task t*. Resources with *Role B* and *Role C* are allocated to *task* t_1 and task t_n, respectively. Hence, if a task among these is executed, the corresponding role will perform the task.

- The organisation allocation specifies that the task has to be executed by the specified department. For example, in Figure 5.1.c the *task t*, *task t_1*, and *task t_n* should be executed by a resource that belongs to *department S*.
- Supervision allocation signifies that a task has to be executed under supervision. For instance, in Figure 5.1.d *task t* is executed by *Role A*, but under supervision of *Role B*.
- Two-role allocation is based on the 4-eye principal where two resources collectively execute a task. *Task t* (in Figure 5.1.e), for instance, is executed by *Role A* and *Role B* together.
- The separation/binding of duties notations can be attached to an atomic task, inclusive task or inclusive control-flow. This way, these semantics specify whether two atomic tasks or two tasks within the inclusive list can be executed by the same resource. An equal sign in the circle represents the binding of duties, and a not-equal sign stresses the separation of duties. For example, in Figure 5.1.f, the *task A* and *B* cannot be executed by the same resource, however, *task C* and *D* can be executed by the same resource.
- The organisation structure represents the relation between the resources, roles and departments. Figure 5.1.g shows an example of an organisation structure. On the one hand, this organisation structure represents which roles are assigned to a specific department. For instance, the resource *Mr. Z* has *Role C*, which is assigned to the *General* department. The *General* department includes the roles which cannot be assigned to one specific department (e.g. a receptionist). On the other hand, it shows which resources will take up a specific role. *Mrs. Y*, for instance, has *Role A* and *B* within the *Department S*.

These resource patterns are simple and straightforward. Though, the combination of the different resource patterns within a small model can lead to complications. For example, the use of several separation/binding of duties constructs for multiple tasks within an inclusive activity or control-flow could lead to complex models. Therefore, the different resource notations should not be overused when it comes to an inclusive activity and control-flow.

6 Evaluation

This research focuses on process models which are used to communicate a process. The use of graphical notations within a process models are meant to facilitate the comprehension of a process leading to effective communication. This section compares the comprehensibility of BPMND+R with BPMN and Declare. For this comparison, a process is modelled using BPMN, Declare and BPMD+R. Next, a two-fold evaluation is conducted where, on the one hand, the 7PMG are applied to the three models (discussed in Section 6.1). On the other hand, an empirical study is executed to find out which of the three models is considered to be the most comprehensible for the participants.

6.1 The process models

Hybrid modelling languages are developed to model processes which contain both structured and unstructured behaviour. A process with these characteristics was found in the work of De Smedt et al. in [16], this process represents the journey of a PhD student. This process description operated as a starting point. Eventually, several alterations were made to the description (e.g. including the resource perspective of the process). Consequently, the following process description was constructed:

During the PhD period, the student creates content which is subsequently published in journals or presented at a conference. Every time the PhD student wants to publish in a journal or present at a conference, he will have to create content again. The content creation is under the supervision of a doctorate holder who also has to supervise the first and the second seminar. The progress of a PhD student throughout his career also contains the strict order of a first and second seminar followed by the defence. The seminars are held to discuss the previously created content with a group of experts. The first seminar cannot happen before a contribution at a conference and the second seminar has to be preceded by a journal publication.

A jury evaluates the journal publications and conference contributions. The jury members who evaluate the conference contribution should be different from the jury members who evaluate the journal paper publication. Finally, the jury of defence should be from an external university (not the university to which the PhD student belongs).

The BPMN, Declare and BPMND+R models representing the described behaviour are constructed. These models can be found in Appendix B to D. Each of these models contains the core constructs of their respective language. In this manner, the models are representative and ensure content validity. For instance, the BPMN model covers the basic control flow patterns (e.g. sequence, exclusive choice, merge, or-choice) as well as loops [81]. Declarative model, in turn, includes all major constraint groups (i.e., existence, relation and negation constraints) [13]. BPMND+R model has fundamental notations of control-flow (e.g. atomic task, inclusive activity, exclusive activity, inclusive control-flow, gateway) and various resource patterns (e.g. separation of duties, supervision allocation). Moreover, these models were built considering the good modelling practices [13]. For instance, every gateway had a text annotation explaining the routing-decision. Declare model was specifically built, using the notations described in [77]. For the declarative model, it was made sure that there were no dead activities, resulting from contradicting constraints [53].

6.2 Theoretical evaluation

For the theoretical evaluation of the model, the 7PMG are used. The guidelines are specifically designed to increase the comprehensibility of a process model. As discussed in Section 3.3.2, different studies provide various thresholds to enforce each of these guidelines within the model. The thresholds define the number of constructs that can be included in the model according to the guideline in consideration. Table 6.1 give the minimum recommended threshold for each guideline throughout the different studies. Each model constructed in the previous section is evaluated according to the stated threshold. The model that surpasses most of these minimum thresholds is considered more complex than the other two.

7PMG
G1. Use maximum 31 nodes.
G2. Use 3 inputs or outputs per connector.
G3. Avoid gateway mismatches.
G4. Use maximum 10 XOR, 7 AND and 4 OR decision nodes.
G5. Avoid OR routing elements.
G6. Use verb-object activity labels.
G7. Use maximum 50 elements.

Table 6.1: Overview of 7PMG thresholds

In Table 6.2, the thresholds are converted into questions. These inquire whether or not a process model exceeds the defined thresholds. If a model does not exceed the threshold, it complies with the guideline; therefore, the answer is *Yes* to the postulated question. If the model does exceed the threshold, it does not comply with the guideline, and therefore the answer is *No* to the defined question. Furthermore, each cell also provides the number of constructs under consideration. For example, guideline 5 is converted into the following question: *Maximum 10 XOR, 7 AND and 4 OR decision nodes?*. The BPMN model does comply with this guideline, as it has 10 XOR, 2 OR and no AND decision nodes in the model. Declare models, however, and do not include any gateways. This guideline is not valid for the Declare model. The BPMND+R has 3 XOR and no OR and AND decision nodes, consequently, it complies with this guideline.

Declare notations do not include gateways; thus, G3 and G4 are not valid for the Declare model. This can be noticed in the fact that the Declare model has the least threshold surpasses. Both BPMN and BPMND+R models exceed in one threshold, but Declare exceeds zero thresholds. Consequently, the 7PMG suggest that BPMN and BPMND+R might be relatively difficult to understand than Declare for their users. However, this conclusion has to be taken carefully as these guidelines are designed for procedural models. For a better evaluation of declarative and hybrid models, another set of guidelines is required.

6.3 Empirical evaluation

To further evaluate the comprehensibility of BPMND+R an empirical study is conducted. This study centres around the following research question: "How

Thresholds	BPMN	Declare	BPMND+R
G1. Maximum 31 nodes?	**Yes** - 23 nodes	**Yes** - 5 nodes	**Yes** - 12 nodes
G2. Maximum 3 input or output arcs per connector?	**No** - 5 arcs	**Yes** - 4 arcs	**Yes** - 2 arcs
G3. Only 1 start and 1 end events?	**Yes** - 1 start 1 end event	**Yes** - 1 start 1 end event	**Yes** - 1 start 1 end event
G4. Gateways mismatches are avoided?	**Yes**	*No Gateways*	**No**
G5. Maximum 10 XOR, 7 AND and 4 OR decision nodes?	**Yes** - 10 XOR 2 OR 0 AND	*No Gateways*	**Yes** - 3 XOR 0 OR 0 AND
G6. Verb-object activity labels?	**Yes**	**Yes**	**Yes**
G7. Maximum 50 elements?	**Yes** - 50 elements	**Yes** - 15 elements	**Yes** - 24 elements

Table 6.2: 7PMG: The evaluation counting

comprehensible is BPMND+R compared to Declare and BPMN?". The following section presents the research design and results of the empirical study, which answers the postulated question.

6.3.1 Experimental design

This experimental design approach requires a population who has adequate knowledge of BPMN and Declare notations. The master students of UHasselt and KU Leuven had a course on Business Process Modelling which included BPMN and Declare notations. These students not only meet the knowledge requirement, but they are also a convenience sample for this academic research. As most of the empirical studies, this study will also be conducted under the master students [23].

In preparation for the experiment, the three process models constructed previously are slightly altered. The changes in the process models are made to ensure that the behaviour of each process model is slightly different from the other two. Besides, for each model several yes/no questions are defined related to the behaviour of the process models. These alterations avoid any learning-aspects during the questionnaire [19]. The modified models are included in Appendix E to G.

The designed questionnaire (included in Appendix H) can be divided into three phases:

1. At the beginning of the questionnaire, the participants are asked to assess their theoretical knowledge on and experience of BPMN and Declare notations. Next, the participants receive a short introduction to the modelled process. The concise description of the process only introduces the activities

and resources within the process. The relations between the activities are not included in the description. This way, the participants are obligated to use the process models for information-seeking.

2. The second phase includes BPMN, Declare and BPMND+R models. At first, the participants receive a short introductory video about BPMN notations that are used within the process model. This ensures that all the participants have the required knowledge to understand the process model. Then, the participants receive the BPMN model and pre-defined yes/no questions. These yes/no questions are used to objectively measure the comprehension of a model, relatively to the other models. During the questionnaire, the participants had access to a concise summary of the constructs used in the model. This way, the participants do not have to memorise or recall the explained constructs [23]. After answering the yes/no questions of the model, the participants are asked to rank the language on a scale of comprehensibility and elaborate on the perceived ease of understanding the model. The introductory video, the yes/no questions and the subjective indicators are then repeated for Declare and BPMND+R models. Due to the technical limitations, there is no randomisation within the three language-related parts of the survey.

3. At last, participants are asked to rank the three languages on the perceived complexity and provide qualitative feedback on the most complex perceived language.

Data collection

The following data is gathered in the survey related to the self-assessment on the knowledge on and experience with BPMN and Declare modelling notations:

– THEORY(BPMN/Declare): Participants make a self-assessment of their theoretical knowledge on BPMN and Declare modelling, on a five points ordinal scale.

– PRACTICE(BPMN/Declare): Participants make a self-assessment of their practical experience with BPMN and Declare modelling, on a four points ordinal scale.

The following objective indicators of comprehension are measured:

– DURATION(BPMN/Declare/BPMND+R): The time that the participants take to answer the yes/no questions of a specific model.

– SCORE: This variable is recorded for each question related to a model. The recorded answer can be correct (value of the variable is 1); the participant did not know the answer (value of the variable is 0); or the answer was incorrect (value of the variable is -1).

- M-SCORE(BPMN/Declare/BPMND+R): This variable captures the sum of the correct answers for each model (SCORE). It serves as an operationalisation of understandability related to a model.

Lastly, several subjective indicators of comprehension are recorded:

- SCALE(BPMN/Declare/BPMND+R): The complexity scale provides the perceived comprehension of the model. The results can differ between Extremely difficult to understand and Extremely easy to understand. This variable is recorded for each language.
- M-FEEDBACK(BPMN/Declare/BPMND+R): This variable provides qualitative feedback on the perceived comprehension of a language. This variable is recorded for each language.
- RANKING: At the end of the questionnaire, the participants rank the three languages regard to their perceived understandability.
- FEEDBACK: The participants also give qualitative feedback on the language that they experienced to be the most difficult.
- GENERAL_FEEDBACK: The participants give optionally general feedback on the questionnaire.

Hypothetical relations between variables and comprehension

Before conducting the statistical analysis, several hypothetical connections between the different variables can be explicated. These hypotheses are based on the work in [49], as the authors also intended to measure the comprehension of the different models. In particular, two hypotheses related to the personal factors and model factors can be defined:

1. The higher the M-SCORE(BPMN/Declare/BPMND+R), the more comprehensible a language is for the participant.
2. A high M-SCORE(BPMN/Declare/BPMND+R) of a model, is connected with lower values in DURATION and higher values in RANKING.

A high score on the yes/no questions regarding the model represents high comprehensibility of that language as it indicates the accuracy of the answers. Therefore, the participants with a high score on a model should efficiently answer the questions (low DURATION) and rank the language high on the comprehensibility scale (high RANKING) [24,49]. In the following section, several statistical methods are used to assess these hypotheses.

6.3.2 Results

The collected data can be divided into two categories. These are discussed and analysed in this section.

Objective indicators

Overall, 18 students filled in the questionnaire. Of these 18 students, on a scale of 1 to 5, nine considered themselves to have a mediocre knowledge on BPMN, and the other nine claimed to have rather strong knowledge on BPMN. Regarding Declare, four students indicated to have weak theoretical knowledge on Declare; the other 14 were evenly distributed between rather weak knowledge on Declare and common theoretical knowledge on Declare. Finally, two of the participants indicated that they use BPMN regularly in their group projects, student jobs, internships or thesis. The rest of the students had never used BPMN (7 students) or sometimes used BPMN in practice (9 students). However, 13 students indicated that they had never used Declare in practice and the other five indicated that they used Declare sometimes in practice. These five students had a group work in which they used the Declare notations once. To sum up, the participants claimed to have a higher knowledge on BPMN than Declare. Nonetheless, the students have rather low experience with using both BPMN and Declare in practice.

The DURATION represents the time that the students attributed to understand the model and solve the related questions. The descriptive statics of this variable are displayed in Table 6.3. The average time taken to solve the yes/no questions for each model is around 8 minutes (BPMN = 9.53; Declare = 8.89; BPMND+R 7.81). The variable does contain extreme outliers for BPMN (44 minutes), Declare (42 minutes) and BPMND+R (28 minutes) model as these values are greater than three times the interquartile range (Q3-Q1). An explanation of these outliers could be that the survey was filled in online. Thus, the students might have engaged in other tasks during the questionnaire. This decreases the credibility of this static, and it is not used in further analysis.

	BPMN	Declare	BPMND+R
Minimum	3.83	1.63	3.14
1st Quartile (Q1)	6.26	3.78	5.05
Median	7.82	6.91	6.43
3rd Quartile (Q3)	9.46	10.23	8.56
Maximum	44.36	42.62	28.43
Interquartile Range(IQR)	3.20	6.45	3.51
Q3 + 3*IQR	19.06	29.58	19.09
Mean	9.53	8.89	7.81
Variance	79.283	85.169	31.095

Table 6.3: Descriptive statistics of DURATION

Next, the M-SCORE of each model is compared to the other. As mentioned earlier, the M-SCORE is the sum of the SCORE variable per model. A SCORE can be 1 when answered correctly; -1 if the answer is wrong; and 0 when the respondent indicates that he does not know the answer. Therefore, when the SCORE is summed for each model, the M-SCORE can also have negative values (mapped on the y-axis of Figure 6.1). The boxplots in Figure 6.1 give a first impression of the scores per model (descriptive statistics in Table 6.4). The median score of BPMN and BPMND+R model (6.00) is equal, while the Declare model (3.00) has a lower median score. The variation of the BPMN score (5.595) is also lower than the other two models. Thus, the results for the BPMN model are more consistent, leading to higher answer accuracy. This can also be seen in Figure 6.1. Moreover, the scores for the BPMN model are consistent on the higher end. Besides, the upper 50% of BPMN and BPMND+R scores have the same distribution. BPMN and Declare model also have an outlier that is not extreme (<Q3 - 1.5*IQR). Each model has one respondent whose score was lower than 25%. From these boxplots and descriptive statistics, it can be concluded that the average scores of BPMN and BPMND+R are higher than Declare scores.

	BPMN	Declare	BPMND+R
Minimum	0	-5	0
Median	6	3	6
Maximum	10	10	10
Interquartile Range(IQR)	3	5	4
Q1 - 1.5*IQR	.5	-5	-2
Q3 - 1.5*IQR	-4	-10	-8
Mean	6.22	3.28	5.67
Variance	5.59	14.33	9.41

Table 6.4: Descriptive statistics of M-SCORE

To analyse whether there are significant differences between the scores of the three models, the statistical two-paired ANOVA test is executed. The ANOVA test is based on following assumptions: 1) the variable is normally distributed 2) the sample cases are independent of each other 3) the variance among the groups is equal 4) the groups have equal sample size [36]. These assumptions are tested for the M-SCORE variable:

1. To test the normality assumption, the Shapiro-Wilk test is chosen. This test is considered to be more appropriate for small sample size (<50) than the Kolmogorov–Smirnov test [70]. The results of the test are given in Table 6.5; the significance level of each modelling language is above 0.05 [71]. According to the Shapiro-Wilk test, the normality assumption holds for the M-SCORE variable.

	BPMN	Declare	BPMND+R
P-value	.181	.908	.244

Table 6.5: Normality test of M-SCORE (Shapiro-Wilk)

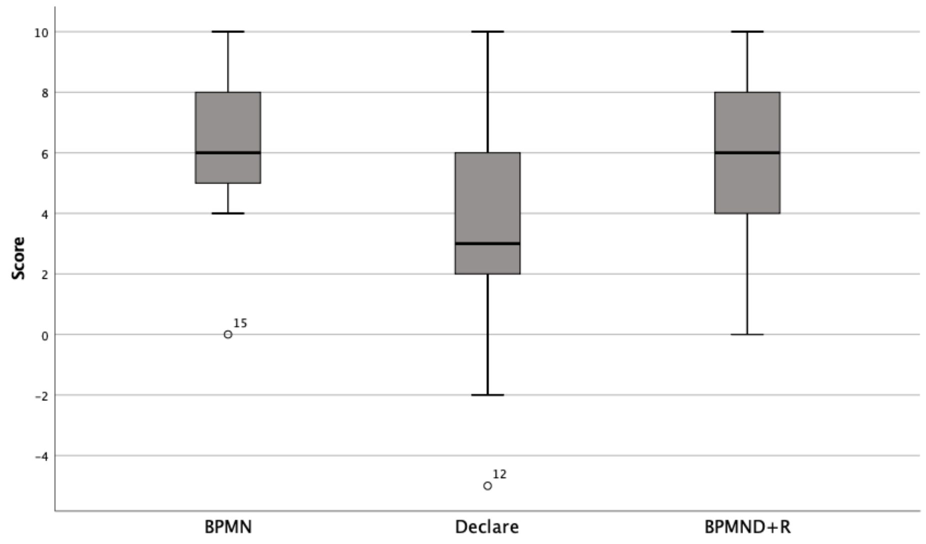


Fig. 6.1: Boxplots of SCORE per model

2. The sample cases are independent as the participants individually filled in the questionnaire.
3. The test of homogeneity of variance is performed to ensure the variance equality. This Leven's static (sig. = .242) is not significant at the level of .05. Thus, the groups have homogeneous variances [41].
4. The sample size of each group is 18.

The ANOVA-assumptions hold for variable M-SCORE. The null and alternative hypothesis for these ANOVA tests is defined as follows:

> H_0: There is no significant difference between the scores of the different models.
> H_1: There is a significant difference between the scores of the different models.

The outcome of the ANOVA statistical analysis has a p-value of 0.016, which is significant for an alpha level of 0.05. Consequently, the null hypothesis can be discarded with a 95% confidence level. Thus, the scores of the different models are significantly different from each other. Furthermore, the Tukey HSD test is conducted to analyse the pairwise differences between the modelling languages [3]. The difference between the mean of each pair of modelling language is shown in Table 6.6. This shows that the average scores of Declare model are significantly lower than the average scores of the BPMN model. However, the difference between the average score of Declare and BPMND+R is not significant at the 95% confidence level (p-value = 0.066).

	BPMN-Declare	Declare-BPMND+R	BPMND+R-BPMN
Difference	2.944*	-2.389	-0.556
P-value	0.018	0.066	0.856

Table 6.6: Tukey's HSD mean comparison for M-SCORE (*significant at 95% confidence level)

After the questions of each model, the students were asked to rank that process modelling language on the perceived difficulty of the model. The participants ranked the model on a scale of 1 to 7, with 1 being extremely difficult and 7 extremely easy to understand. The perceived difficulty, as indicated by the participants, is shown in Figure 6.2. The descriptive statics of the variable are given in Table 6.7. The average value for BPMN, Declare and BPMND+R are 3.94, 2.61 and 4.56, respectively. There are several notable differences, i.e., the most significant one being that Declare was the most difficult to understand for the users as it has the lowest mean with a variance of 1.08. Thus, the results are grouped at the lower side of the scale. The majority of the participants found BPMND+R slightly easy to understand (highest average). BPMN rankings are evenly distributed on the scale.

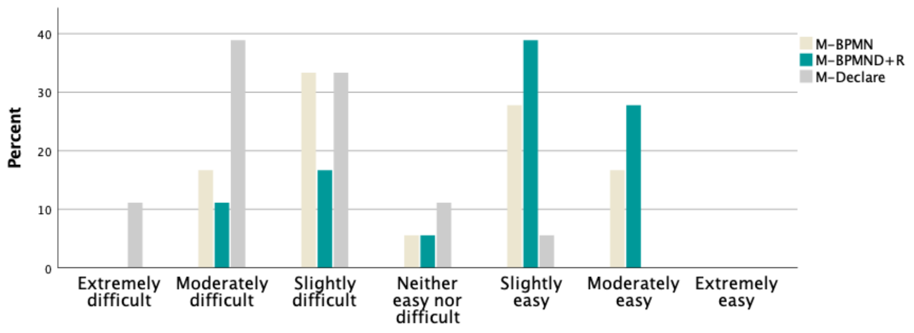


Fig. 6.2: Percentage of perceived difficulty scores of the models relative to the total number of participants

At the end of the questionnaire, the students were asked to rank the process modelling language according to their complexity. Majority of the students reported Declare as the most difficult language to understand, while BPMND+R was the easiest to understand.

Subjective indicators

To validate the results further, the participants are asked to provide qualitative feedback on the comprehensibility of each language. This feedback is included in Appendix I. Nine participants found BPMN hard to understand because the

	BPMN	Declare	BPMND+R
Minimum	2	1	2
Median	3.5	2.5	5
Maximum	6	5	6
Mean	3.94	2.61	4.56
Variance	2.06	1.08	1.90

Table 6.7: Descriptive statistics RANKING

flow of the model had too many possibilities, and it gave too much information. A large number of arcs, gateways and loops within the process, made it harder for the participants to understand which behaviour is exactly allowed. Though, there are participants, who found the control-flow of the BPMN model quite clear as it only included five activities and textual descriptions. These textual descriptions provided additional information about the resources of the process. One participant also pointed out that the names of the tasks were very clear and textual conditions on XOR gateways made the flow of the process model easy to understand.

The majority of the participants (15) found Declare model rather difficult to understand. The Declare model included too many constraints of which the behaviour and notations could not always be precisely differentiated. As the model lacks a sequential flow, the participants experienced additional difficulties. The participants mentioned that they had to understand each constraint, interpret it within the model and finally link it with other constraints and activities. On the one hand, several participants argued that there were too many possibilities and freedom within the model. On the other hand, a few participants said that they could focus on the individual relations of two activities which made the model easier to understand than BPMN. Finally, several participants suggested that textual notations within the process model might make the Declare model more understandable and provide more information on the resource perspective of the process.

Most of the participants experienced BPMND+R as a rather easy modelling language. The participants appreciated the clear notations of the resource perspective, which were not included in the previous two models. Many participants mentioned that once they get more familiar with BPMND+R notations, they might understand the process model better. Other argued that the notations were quite obvious and clear after the short introductory video. For some participants, the notation of exclusive control-flow ($EX(t_1, \ldots t_n)$) was not quite clear. One participant also mentioned that within the BPMND+R notations, both the arcs and activities could represent tasks, which made the model slightly difficult to understand. Finally, the participants mentioned that BPMND+R notations provided a middle-ground for BPMN's strict control-flows and Declare's loosely constraints.

In addition, within the general remarks, several participants appreciated the existence of constraint notations within the Declare and BPMND+R models. These constraints made it easier to understand how many times an activity could be executed.

Lastly, the participants were asked to give feedback on one language which they perceived the most difficult to understand. Almost every participant mentioned Declare as the most difficult language to understand because of the abundant constraints and their similar notations. A few also mentioned BPMN as difficult because it included many loops with a large number of gateways. None of the participants mentioned BPMND+R as the most complex language.

Both the objective and subjective indicators conclude that the participants experienced the BPMND+R and BPMN models to be more comprehensible than Declare. This conclusion is also validated by the qualitative feedback. The subjective indicators suggest that the BPMND+R model was slightly easier to understand compared to BPMN and Declare model. It is also apparent that the BPMND+R notations are clear and evident, despite that these notations were new to the participants compared to the BPMN and Declare notations. Nevertheless, several participants mentioned that it was harder to differentiate between inclusive/exclusive activities.

Nonetheless, the results of this empirical study have to be treated cautiously and considered in the context of several limitations. The subjects of theses studies were 18 students that were convenient to reach for the researcher. Besides only a few participants claimed to have an experience with BPMN and Declare. This limits the generalisability of the results. The motivation of participants and their learning techniques might not be representative of the whole population as no professional modellers were included in this research. Nevertheless, the use of student as subjects is a well-established practice of experimental studies [67]. Moreover, the process models used in this language represented the basic concepts of that language, yet there was only one model presented for each modelling language. This limits the representability of used modelling languages.

7 Conclusion

When developing a process model for communication purposes, a multitude of process modelling languages can be used. Traditionally, a distinction is made between procedural and declarative languages. Procedural languages support structured processes, and declarative languages are used for flexible processes. Recently, researchers recognised that procedural and declarative languages should not be seen as mutually exclusive. This brings the possibility of the hybrid process modelling languages. The combined notations of procedural and declarative languages provide a more practical solution for structured and unstructured processes. However, hybrid languages are not fully developed yet. On the one hand, they lack the graphical notations to represent the resource perspective of a process. On the other hand, there are no empirical studies that evaluate the comprehensibility of these languages.

This paper presented and evaluated a resource-aware hybrid modelling language, BPMND+R. This language is specifically designed to represent the resource patterns in a comprehensible way. BPMND+R is evaluated in a theoretical and empirical study by comparing it to BPMN and Declare. The theoretical study evaluated the Declare model to be more comprehensible than the BPMN and BPMND+R models. Due to the lack of an overall set of easy-to-use guidelines for procedural and declarative languages, the models were evaluated using guidelines built for procedural models. Consequently, the results of this study remain questionable. The results from the empirical study suggested that the Declare model was more complex for the participants than the BPMND and BPMND+R models. Overall, the participants stated BPMND+R as straightforward and easy to understand. Other participants found the BPMND+R control-flow notations ambiguous and complex.

Possible directions for future research follow from the limitations of theoretical and empirical studies. Firstly, future work can conduct an empirical study that evaluates the comprehensibility of this hybrid language with a larger sample that also includes professional modellers and domain experts. Secondly, research efforts can be directed to introduce easy-to-use guidelines applicable to declarative and hybrid models. Finally, this research only visualises simple resource patterns within a hybrid model. Besides, the used graphical notations are language-independent. Future work can be conducted to extend other hybrid languages and visualise more complex resource-related relations within a process.

References

1. Van der Aalst, W.M.: Formalization and verification of event-driven process chains. Information and Software technology **41**(10), 639–650 (1999)
2. van der Aalst, W.M., Adams, M., ter Hofstede, A.H., Pesic, M., Schonenberg, H.: Flexibility as a service. In: International Conference on Database Systems for Advanced Applications. pp. 319–333. Springer (2009)
3. Abdi, H., Williams, L.J.: Tukey's honestly significant difference (hsd) test. Encyclopedia of Research Design. Thousand Oaks, CA: Sage pp. 1–5 (2010)
4. Adams, M., Ter Hofstede, A.H., Edmond, D., Van Der Aalst, W.M.: Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 291–308. Springer (2006)
5. Aguilar-Saven, R.S.: Business process modelling: Review and framework. International Journal of production economics **90**(2), 129–149 (2004)
6. Andaloussi, A., Burattin, A., Slaats, T., Kindler, E., Weber, B.: On the declarative paradigm in hybrid business process representations: A conceptual framework and a systematic literature study. Information Systems **91**, 101505 (07 2020)
7. Becker, J., Rosemann, M., Von Uthmann, C.: Guidelines of business process modeling. In: Business process management, pp. 30–49. Springer (2000)
8. Bidgoly, A.J., Ladani, B.T.: Trust modeling and verification using colored petri nets. In: 2011 8th International ISC Conference on Information Security and Cryptology. pp. 1–8. IEEE (2011)
9. Booch, G.: The unified modeling language user guide. Pearson Education India (2005)
10. Bravetti, M., Núñez, M., Zavattaro, G.: Web Services and Formal Methods: Third International Workshop, WS-FM 2006, Vienna, Austria, September 8-9, 2006, Proceedings. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2006), https://books.google.be/books?id=ENZtCQAAQBAJ
11. Byrne, M., Keary, E., Lawton, A.: How to conduct a literature review (2012)
12. Chinosi, M., Trombetta, A.: Bpmn: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012)
13. Corradini, F., Ferrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., Spagnolo, G.O.: A guidelines framework for understandable bpmn models. Data & Knowledge Engineering **113**, 129–154 (2018)
14. De Giacomo, G., Dumas, M., Maggi, F.M., Montali, M.: Declarative process modeling in bpmn. In: International Conference on Advanced Information Systems Engineering. pp. 84–100. Springer (2015)
15. De Smedt, J., De Weerdt, J., Serral, E., Vanthienen, J.: Improving understandability of declarative process models by revealing hidden dependencies. In: International Conference on Advanced Information Systems Engineering. pp. 83–98. Springer (2016)
16. De Smedt, J., De Weerdt, J., Vanthienen, J.: Fusion miner: Process discovery for mixed-paradigm models. Decision Support Systems **77**, 123–136 (2015)
17. De Smedt, J., De Weerdt, J., Vanthienen, J., Poels, G.: Mixed-paradigm process modeling with intertwined state spaces. Business & Information Systems Engineering **58**(1), 19–29 (2016)
18. Debois, S., Hildebrandt, T.T., Marquard, M., Slaats, T.: Hybrid process technologies in the financial sector. In: BPM (Industry track). pp. 107–119 (2015)
19. Delgado-Rico, E., Carretero-Dios, H., Ruch, W.: Content validity evidences in test development: An applied perspective. International Journal of Clinical and Health Psychology España **12**(3), 449–460 (2012)

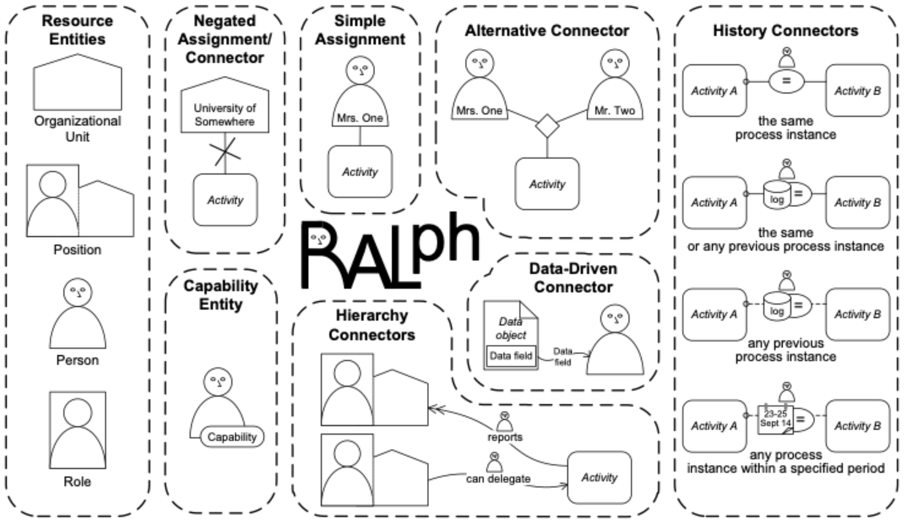
20. Dumas, M., La Rosa, M., Mendling, J., Mäesalu, R., Reijers, H.A., Semenenko, N.: Understanding business process models: the costs and benefits of structuredness. In: International Conference on Advanced Information Systems Engineering. pp. 31–46. Springer (2012)
21. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)
22. Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of understandability. In: Enterprise, Business-Process and Information Systems Modeling, pp. 353–366. Springer (2009)
23. Figl, K.: Comprehension of procedural visual business process models. Business & Information Systems Engineering **59**(1), 41–67 (2017)
24. Figl, K., Mendling, J., Strembeck, M., Recker, J.: On the cognitive effectiveness of routing symbols in process modeling languages. In: International Conference on Business Information Systems. pp. 230–241. Springer (2010)
25. Giannakopoulou, D., Havelund, K.: Automata-based verification of temporal properties on running programs. In: Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001). pp. 412–416. IEEE (2001)
26. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: International Conference on Business Process Management. pp. 5–14. Springer (2006)
27. Goedertier, S., Vanthienen, J., Caron, F.: Declarative business process modelling: principles and modelling languages. Enterprise Information Systems **9**(2), 161–185 (2015)
28. Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding declare models: strategies, pitfalls, empirical results. Software & Systems Modeling **15**(2), 325–352 (2016)
29. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly pp. 75–105 (2004)
30. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Nested dynamic condition response graphs. In: International conference on fundamentals of software engineering. pp. 343–350. Springer (2011)
31. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., et al.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: International Workshop on Web Services and Formal Methods. pp. 1–24. Springer (2010)
32. Indulska, M., Recker, J., Rosemann, M., Green, P.: Business process modeling: Current issues and future challenges. In: International Conference on Advanced Information Systems Engineering. pp. 501–514. Springer (2009)
33. Jablonski, S., Bussler, C.: Workflow management: Modeling concepts. Architecture and Implementation. Thomson Computer Press **41**(43), 112 (1996)
34. Johannesson, P., Perjons, E.: An introduction to design science. Springer (2014)
35. Kesari, M., Chang, S., Seddon, P.B.: A content-analytic study of the advantages and disadvantages of process modelling. ACIS 2003 Proceedings p. 2 (2003)
36. Khan, A., Rayner, G.D.: Robustness to non-normality of common tests for the many-sample location problem. Advances in Decision Sciences **7**(4), 187–206 (2003)
37. Kirschner, P.A.: Cognitive load theory: Implications of cognitive load theory on the design of learning (2002)
38. Knolmayer, G., Endl, R., Pfahrer, M.: Modeling processes and workflows by business rules. In: Business Process Management, pp. 16–29. Springer (2000)

39. Koschmider, A., Figl, K., Schoknecht, A.: A comprehensive overview of visual design of process model element labels. In: International Conference on Business Process Management. pp. 571–582. Springer (2016)
40. Leopold, H., Mendling, J., Günther, O.: Learning from quality issues of bpmn models from industry. IEEE Software **33**(4), 26–33 (2015)
41. Levene, H.: Robust tests for the equality of variance in olkin i., editor.(ed.), contributions to probability and statistics (pp. 278–292). paolo alto (1960)
42. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: From texts to declarative processes and back. BPM (Dissertation/Demos/Industry) **2196**, 66–70 (2018)
43. Lu, R., Sadiq, S., Governatori, G.: On managing business processes variants. Data & Knowledge Engineering **68**(7), 642–664 (2009)
44. Mangan, P., Sadiq, S.: On building workflow models for flexible processes. In: Australian Computer Science Communications. vol. 24, pp. 103–109. Australian Computer Society, Inc. (2002)
45. Marin, M.A.: Introduction to the case management model and notation (cmmn). arXiv preprint arXiv:1608.05011 (2016)
46. Marquard, M., Shahzad, M., Slaats, T.: Web-based modelling and collaborative simulation of declarative processes. In: International Conference on Business Process Management. pp. 209–225. Springer (2016)
47. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7pmg). Information and Software Technology **52**(2), 127–136 (2010)
48. Mendling, J., Sánchez-González, L., García, F., La Rosa, M.: Thresholds for error probability measures of business process models. Journal of Systems and Software **85**(5), 1188–1197 (2012)
49. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: International Conference on Business Information Systems. pp. 142–153. Springer (2008)
50. Mertens, S., Gailly, F., Poels, G.: Towards a decision-aware declarative process modeling language for knowledge-intensive processes. Expert Systems with Applications **87**, 316–334 (2017)
51. Mili, H., Tremblay, G., Jaoude, G.B., Lefebvre, É., Elabed, L., Boussaidi, G.E.: Business process modeling languages: Sorting through the alphabet soup. ACM Computing Surveys (CSUR) **43**(1), 1–56 (2010)
52. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
53. Pesic, M.: Constraint-based Workflow Management Systems: Shifting Control to Users. Ph.D. thesis, Industrial Engineering & Innovation Sciences (01 2008)
54. Pesic, M., Van der Aalst, W.M.: A declarative approach for flexible business processes management. In: International conference on business process management. pp. 169–180. Springer (2006)
55. Pesic, M., Schonenberg, H., Van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). pp. 287–287. IEEE (2007)
56. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: An empirical investigation. In: International Conference on Business Process Management. pp. 383–394. Springer (2011)
57. Reichert, M., Rinderle, S., Dadam, P.: Adept workflow management system. In: International Conference on Business Process Management. pp. 370–379. Springer (2003)

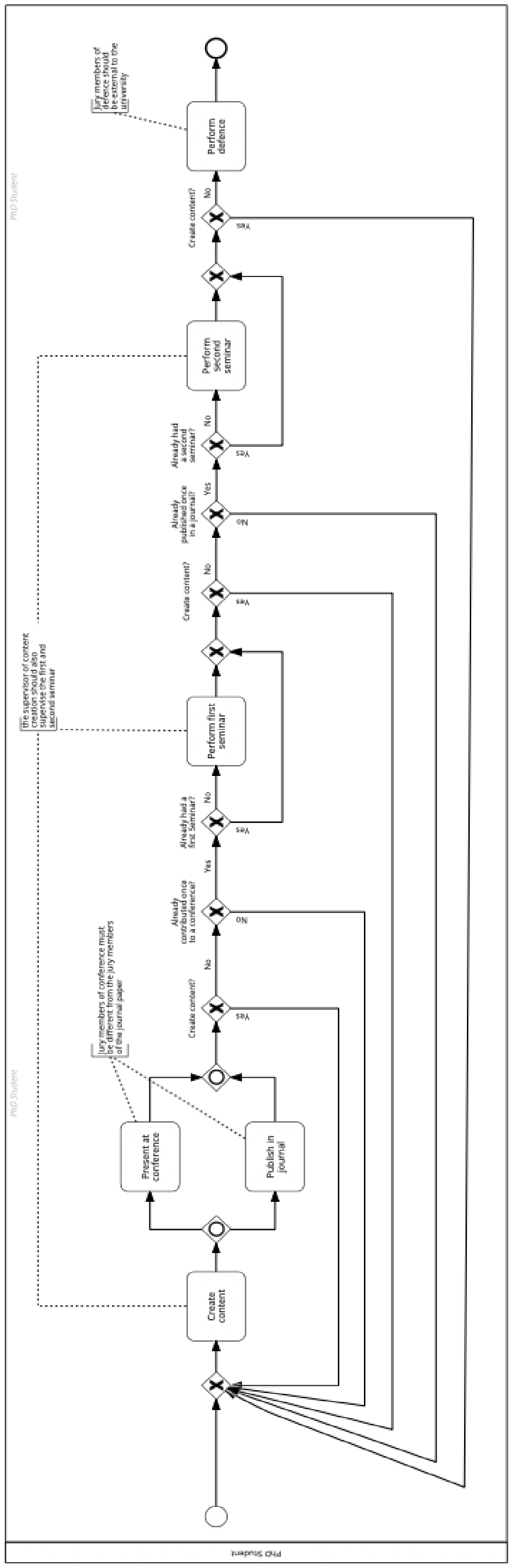
58. Reijers, H.A., van der Aalst, W.M.: Mining hybrid business process models: A quest for better precision. In: Business Information Systems: 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings. vol. 320, p. 190. Springer (2018)
59. Reijers, H.A., Slaats, T., Stahl, C.: Declarative modeling–an academic dream or the future for bpm? In: Business Process Management, pp. 307–322. Springer (2013)
60. Rosemann, M., Recker, J., Indulska, M., Green, P.: A study of the evolution of the representational capabilities of process modeling grammars. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering. pp. 447–461. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
61. Russell, N., van der Aalst, W.M., Ter Hofstede, A.H., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: International Conference on Advanced Information Systems Engineering. pp. 216–232. Springer (2005)
62. Sadiq, S., Sadiq, W., Orlowska, M.: Pockets of flexibility in workflow specification. In: International Conference on Conceptual Modeling. pp. 513–526. Springer (2001)
63. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Information Systems **30**(5), 349–378 (2005)
64. Sánchez-González, L., García, F., Mendling, J., Ruiz, F.: Quality assessment of business process models based on thresholds. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 78–95. Springer (2010)
65. Sánchez-González, L., García, F., Ruiz, F., Mendling, J.: Quality indicators for business process models from a gateway complexity perspective. Information and Software Technology **54**(11), 1159–1174 (2012)
66. Sandkuhl, K., Wiebring, J.: Experiences from selecting a bpm notation for an enterprise. In: International Conference on Business Information Systems. pp. 126–138. Springer (2015)
67. Sarshar, K., Loos, P.: Comparing the control-flow of epc and petri net from the end-user perspective. In: International Conference on Business Process Management. pp. 434–439. Springer (2005)
68. Schönig, S., Jablonski, S.: Comparing declarative process modelling languages from the organisational perspective. In: International Conference on Business Process Management. pp. 17–29. Springer (2016)
69. Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: Enterprise, Business-Process and Information Systems Modeling, pp. 48–63. Springer (2014)
70. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**(3/4), 591–611 (1965)
71. Siegel, S.: Nonparametric statistics. The American Statistician **11**(3), 13–19 (1957)
72. Slaats, T., Schunselaar, D.M., Maggi, F.M., Reijers, H.A.: The semantics of hybrid process models. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 531–551. Springer (2016)
73. of Technology, E.U.: DECLARE Manual. TU/e
74. Ter Hofstede, A.H., van der Aalst, W.M.: Yawl: yet another workflow language. Information Systems **30**(4), 245–275 (2005)
75. (patrick Van Bommel, P.: Qomo: A modelling process quality framework based on sequal. In: Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'07), held in conjunctiun with the 19th

Conference on Advanced Information Systems (CAiSE'07. pp. 118–127. Tapir Academic Press (2007)
76. Van Der Aalst, W.M., La Rosa, M., Santoro, F.M.: Don't forget to improve the process! Business process management **58**(1), 1–6 (2016)
77. Van Der Aalst, W.M., Pesic, M.: Decserflow: Towards a truly declarative service flow language. In: International Workshop on Web Services and Formal Methods. pp. 1–23. Springer (2006)
78. Wang, W., Indulska, M., Sadiq, S., Weber, B.: Effect of linked rules on business process model understanding. In: International conference on business process management. pp. 200–215. Springer (2017)
79. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The declarative approach to business process execution: An empirical test. In: International Conference on Advanced Information Systems Engineering. pp. 470–485. Springer (2009)
80. Westergaard, M., Slaats, T.: Mixing paradigms for more comprehensible models. In: Business Process Management, pp. 283–290. Springer (2013)
81. White, S.A.: Introduction to bpmn. Ibm Cooperation **2**(0) (2004)
82. Wolfswinkel, J.F., Furtmueller, E., Wilderom, C.P.: Using grounded theory as a method for rigorously reviewing literature. European journal of information systems **22**(1), 45–55 (2013)
83. Zugal, S., Pinggera, J., Weber, B.: Toward enhanced life-cycle support for declarative processes. Journal of Software: Evolution and Process **24**(3), 285–302 (2012)

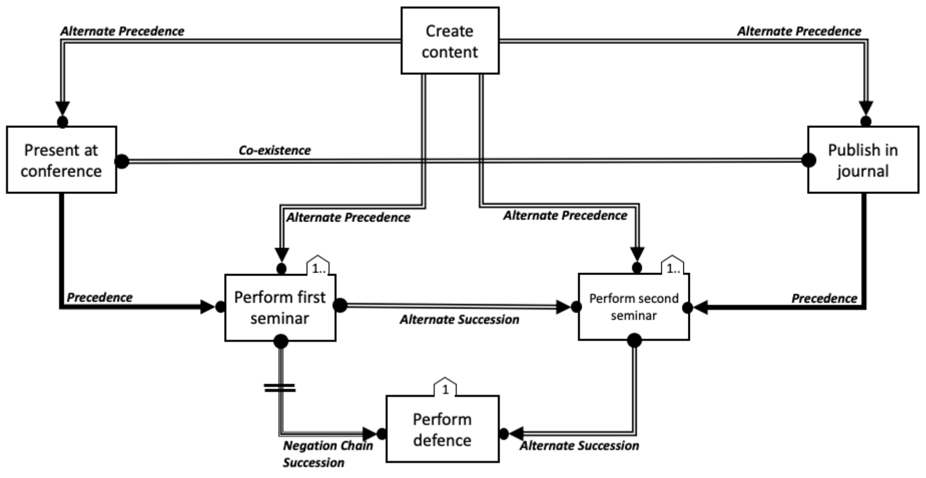
Appendix A The RALph notations



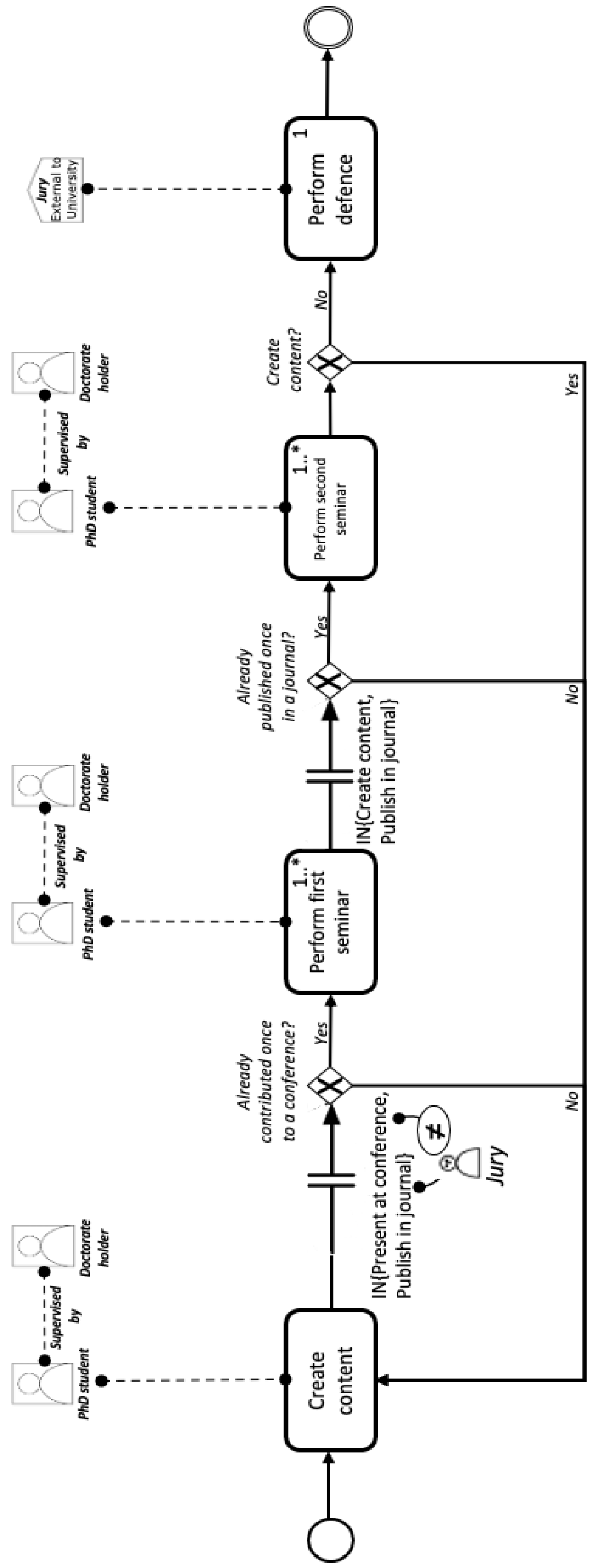
Appendix B BPMN model used for theoretical evaluation


PhD Student
Create content
Present at conference
Publish in journal
Jury members of conference must be different from the jury members of the journal paper
Create content?
Yes
No
Already contributed once to a conference?
Already had a first Seminar?
Perform first seminar
The supervisor of content creation should also supervise the first and second seminar
Already published once in a journal?
Already had a second seminar?
Perform second seminar
Perform defence
Jury members of defence should be external to the university

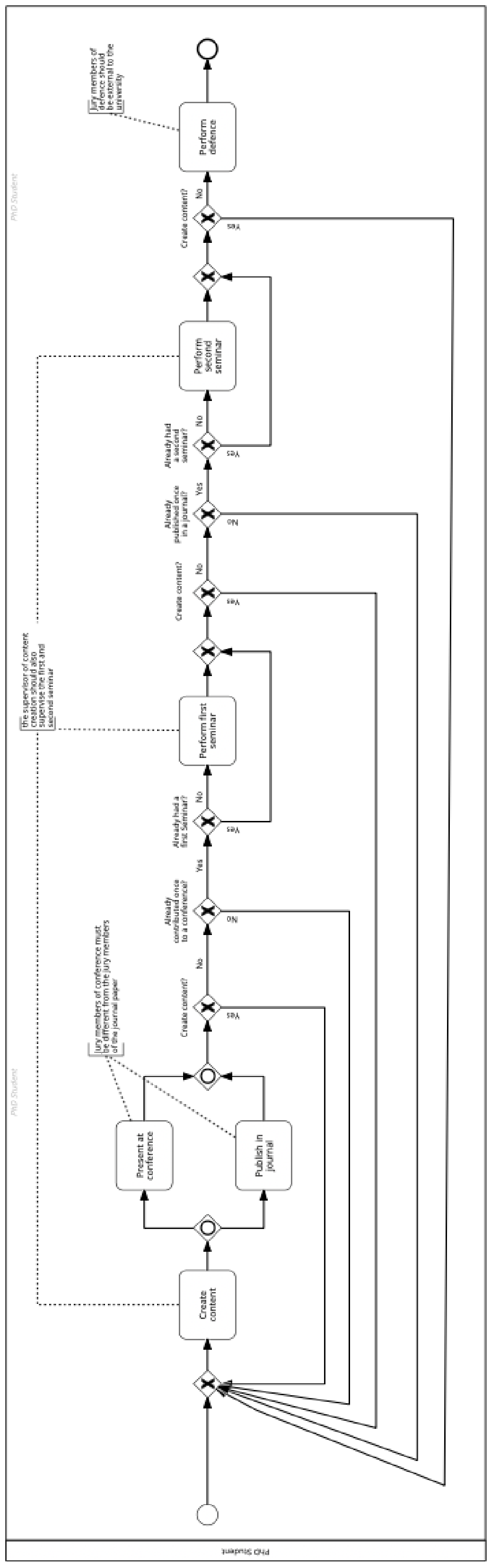
Appendix C Declare model used for theoretical evaluation



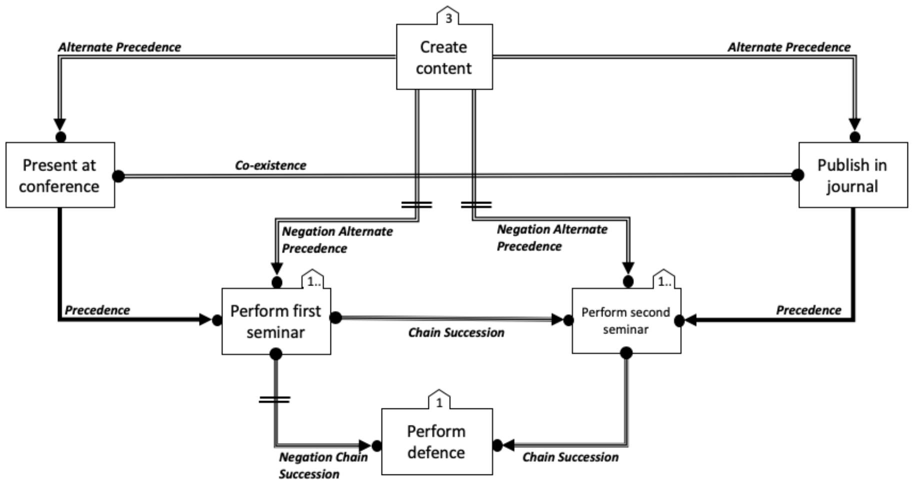
Appendix D BPMND+R model used for theoretical evaluation


Create content
PhD student
Supervised by
Doctorate holder
IN{Present at conference, Publish in journal}
≠
Jury
Already contributed once to a conference?
Yes
No
Perform first seminar
1..*
IN{Create content, Publish in journal}
Already published once in a journal?
Yes
No
Perform second seminar
1..*
Create content?
Yes
No
Perform defence
1
Jury External to University

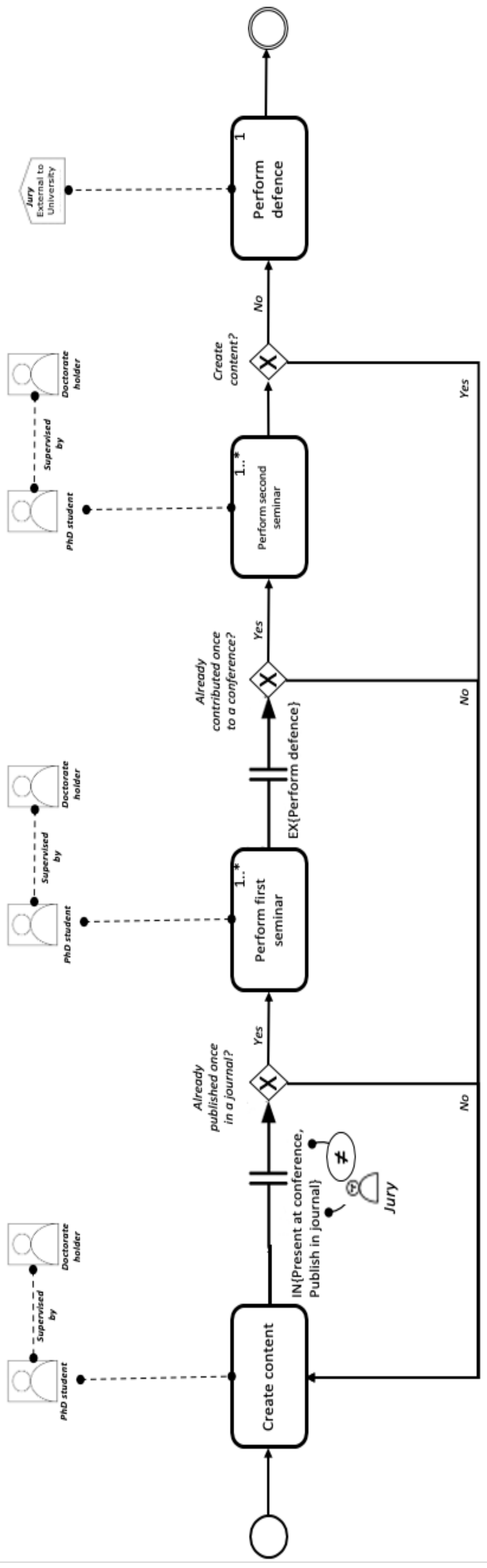
Appendix E BPMN model used for empirical evaluation


PhD Student
Create content
Present at conference
Publish in journal
Jury members of conference must be different from the jury members of the journal paper
Create content?
Yes
No
Already contributed once to a conference?
Yes
No
Already had a first Seminar?
Yes
No
Perform first seminar
the supervisor of content creation should also supervise the first and second seminar
Create content?
Yes
No
Already published once in a journal?
Yes
No
Already had a second seminar?
Yes
No
Perform second seminar
Create content?
Yes
No
Perform defence
Jury members of defence should be external to the university

Appendix F Declare model used for empirical evaluation



Appendix G BPMND+R model used for empirical evaluation



Appendix H Questionnaire

H.1 General Questions-1

Background
Did you follow a course which included BPMN? (Yes/No)
Did you follow a course which included DECLARE? (Yes/No)

Theoretical knowledge on and experience with BPMN
How do you assess your theoretical knowledge on BPMN? (I have weak theoretical knowledge on BPMN/ I have rather weak theoretical knowledge on BPMN/ I have mediocre theoretical knowledge on BPMN/ I have rather strong theoretical knowledge on BPMN/ I have strong theoretical knowledge on BPMN)
How often do you use BPMN in practice (e.g. for group projects, student jobs, internship, thesis, personal use)? (I never use BPMN in practice/ I sometimes use BPMN in practice/ I regularly use BPMN in practice, but not every day/ I use BPMN in practice every day)

Theoretical knowledge on and experience with Declare

How do you assess your theoretical knowledge on business process modelling? (I have weak theoretical knowledge on DECLARE/ I have rather weak theoretical knowledge on DECLARE/ I have mediocre theoretical knowledge on DECLARE/ I have rather strong theoretical knowledge on DECLARE/ I have strong theoretical knowledge on DECLARE)

How often do you use DECLARE in practice (e.g. for group projects, student jobs, internship, thesis, personal use)? (I never use DECLARE in practice/ I sometimes use DECLARE in practice/ I regularly use DECLARE in practice, but not every day/ I use DECLARE in practice every day)

H.2 Questions regarding BPMN model

Multiple-choice questions related to the model 1. Can you present at a conference and also publish in a journal after only one content creation? (Yes/No/It is not clear form the model/ I don't know)

2. After the second seminar, are you obligated to immediately defend your content that you created during your PhD? (Yes/No/It is not clear form the model/ I don't know)

3. Before the first seminar, can you present at a conference without publishing in a journal? (Yes/No/It is not clear form the model/ I don't know)

4. Can you invite Mr Robert from the University of Chicago as a jury member of more than one conference? (You are doing your PhD at the University of Hasselt) (Yes/No/It is not clear form the model/ I don't know)

5. Can you defend your content, if you only had a journal publication and a conference? (Yes/No/It is not clear form the model/ I don't know)

6. Can you have as many content creations as you want? (Yes/No/It is not clear form the model/ I don't know)

7. Is Mr Steven (your supervisor) going to be in the jury of defence? (Yes/No/It is not clear form the model/ I don't know)

8. After the second seminar, can you directly publish in a journal or present at a conference? (Yes/No/It is not clear form the model/ I don't know)

9. Before the second seminar, you are obligated to have a journal publication, but can you also present at another conference? (Yes/No/It is not clear form the model/ I don't know)

10. Can the jury members of the journal publication also attend the second seminar? (Yes/No/It is not clear form the model/ I don't know)

Model comprehension with Qualitative feedback

How would you rate the BPMN model on a scale of comprehensibility? (Extremely easy/ Moderately easy/ Slightly easy/ Neither easy nor difficult/ Slightly difficult/ Moderately difficult/ Extremely difficult)

What made the BPMN model easy or difficult to understand?

H.3 Questions regarding Declare model

Multiple-choice questions related to the model

1. Can you present at a conference and also publish in a journal after only one content creation? (Yes/No/It is not clear form the model/ I don't know)

2. Is Mr Steven (your supervisor) going to be in the jury of defence? (Yes/No/It is not clear form the model/ I don't know)

3. After the second seminar, can you directly publish in a journal or present at a conference? (Yes/No/It is not clear form the model/ I don't know)

4. Between the first and second seminar, can you publish in a journal? (Yes/No/It is not clear form the model/ I don't know)

5. Can you have as many content creations as you want? (Yes/No/It is not clear form the model/ I don't know)

6. Can you invite Mr Robert from the University of Chicago as a jury member of more than one conference? (You are doing your PhD at the University of Hasselt) (Yes/No/It is not clear form the model/ I don't know)

7. If you want to give the first seminar twice, can you create new content between these two seminars? (Yes/No/It is not clear form the model/ I don't know)

8. Before the first seminar, you are obligated to present at a conference, but are you also obligated to have a journal publication? (Yes/No/It is not clear form the model/ I don't know)

9. Do the jury members of the conference also have to attend the first seminar? (Yes/No/It is not clear form the model/ I don't know)

10. After the second seminar, are you obligated to immediately defend your content that you created during your PhD? (Yes/No/It is not clear form the model/ I don't know)

Model comprehension with qualitative feedback

How would you rate the DECLARE model on a scale of comprehensibility? (Extremely easy/ Moderately easy/ Slightly easy/ Neither easy nor difficult/ Slightly difficult/ Moderately difficult/ Extremely difficult)

What made the DECLARE model easy or difficult to understand?

H.4 Questions regarding BPMND+R model

Multiple-choice questions related to the model

1. Do the jury members of the conference also have to attend the first seminar? (Yes/No/It is not clear form the model/ I don't know)

2. Can you skip the journal publication, if you have already presented at a conference? (Yes/No/It is not clear form the model/ I don't know)

3. Suppose that you created content one time, had a journal publication and a conference, can you now give the first seminar and the second seminar as well? (Yes/No/It is not clear form the model/ I don't know)

4. Before the first seminar, can you present at a conference without publishing in a journal? (Yes/No/It is not clear form the model/ I don't know)

5. If you want to give the first seminar twice, can you create new content between these two seminars? (Yes/No/It is not clear form the model/ I don't know)

6. Is Mr Steven (your supervisor) going to be in the jury of defence? (Yes/No/It is not clear form the model/ I don't know)

7. Between the first and second seminar, can you publish in a journal? (Yes/No/It is not clear form the model/ I don't know)

8. Can you invite Mr Robert from the University of Chicago as a jury member of more than one conference? (You are doing your PhD at the University of Hasselt) (Yes/No/It is not clear form the model/ I don't know)

9. Can you have as many content creations as you want? (Yes/No/It is not clear form the model/ I don't know)

10. Can you present at multiple conferences before having a second seminar?(Yes/No/It is not clear form the model/ I don't know)

Model comprehension with qualitative feedback

How would you rate the BPMND+R model on a scale of comprehensibility? (Extremely easy/ Moderately easy/ Slightly easy/ Neither easy nor difficult/ Slightly difficult/ Moderately difficult/ Extremely difficult)

What made the BPMND+R model easy or difficult to understand?

H.5 Ranking the modelling languages

Which of the following two languages was more difficult to understand?(Declare/BPMN)

Which of the following two languages was more difficult to understand? (Declare/BPMND+R)

Which of the following two languages was more difficult to understand? (BPMN/BPMND+R)

H.6 Comprehension qualitative feedback

Which language was the most difficult to understand and why?

Is there anything I should take into consideration (in general)?

Appendix I Subjective feedback

What made the BPMN model easy or difficult to understand?(You can also answer in Dutch)	**What made the DECLARE model easy or difficult to understand?(You can also answer in Dutch)**	**What made the BPMND+R model easy or difficult to understand?(You can also answer in Dutch)**	**Which language was the most difficult to understand and why?(You can also answer in Dutch)**
Makkelijk om te begrijpen	Onvoldoende kennis van Declare om te kunnen onthouden wat alle verschillende pijlen betekenen	Zeker makkelijker te begrijpen dan Declare, maar ik had toch nog onvoldoende kennis om het model helemaal te snappen. Mits ik meer tijd zou hebben om te leren hoe het werkt, lijkt me dit wel een mooi model	Declare is verwarrend met de hoeveelheid aan verschillende pijlen, is voor mij de minst duidelijke
De video legde duidelijk uit wat de verschillende basiselementen zijn van BPMN en het model ging ook niet verder dan deze basiselementen. Soms was er wel te weinig info om een vraag op te lossen.	DECLARE op zich maakte het wat moeilijk. Je moet meerdere keren terug kijken naar de beschrijving van de verschillende constraints om te weten wat ze precies betekenen.	DECLARE geeft meer info aan BPMN, dus dat is altijd mooi meegenomen, maar het brengt niet extra verwarring in het hele verhaal. De constraints die van DECLARE uitkomen zijn niet zo uitbundig als DECLARE zelf en zijn dus makkelijk te onthouden na een korte uitleg (zoals de video in deze study).	DECLARE, al de betekenissen van de verschillende constraints proberen te onthouden kan wat moeilijk zijn. Ik heb al een vak gehad waarbij DECLARE in voorkwam, maar zelfs na de video kon ik niet alle constraints onthouden en hun betekenissen.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
The clear flows and connections. Also the textual descriptions make it easier to understand.	The lack of textual descriptions and extra annotations made the declare model difficult to understand. Also the different symbols that look very much alike make it different to understand without the declare constraints discription.	The links to people who have to execute and supervise a task make it very clear. However, it is not totally clear which activities you can do in between other activities with an excluding list.	The DECLARE language was the most difficult to understand as it used symbols I had the least experience with. The symbols really look alike and the discription was needed to go through all the activities and flows. There was less descriptive information present in the model and some information like for example the supervisor information seemed to be missing from the model.
Je ziet een flow. De mogelijkheden van volgende stappen zijn duidelijk.	Er kunnen geen notities bijgezet worden.Wel is duidelijk wat moet en wat niet mag.	Moeilijk omdat het nieuw is en er meer verschillende symbolen en figuren bij komen kijken.Als het ingewerkt is, lijkt me dit wel makkelijker en vooral duidelijker dan de andere modelleringen.	declare:omslachtig modelgeen info over resources

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
Too much possibilities (edges)	Too much possibilities which were not in the model explicitly	Easy as there are not too many possibilities to choose from (like the BMPN figure but) but enough to understand it better than declare (declare gave too much freedom).However, there are a lot of symbols to focus on. Both declare and BPMN have not that many symbols so it is hard to focus on them all in this combo.	BPMN. bij dit model waren er veel te veel mogelijkheden om te volgen
All constraints are modeled	Hard to understand the flow	The flow was more clear. The constraints were rather understandable. Should be doable after seeing it in a course and having some experience with the add-on	Declare, since it lacks an easy to follow flow and the given process did clearly have a sort of flow
Er zaten veel pijlen en veel beslissingen in het model waardoor het niet altijd overzichtelijk was	Het is niet altijd duidelijk in welke volgorden bepaalde taken uitgevoerd kunnen/moeten worden. Ook mis ik hier de geschreven opmerkingen (zoals die bij BPMN) die soms cruciale extra informatie kunnen bieden.	Het is de eerste keer dat ik van dit model hoor dus het was niet gemakkelijk om alle elementen direct juist toe te passen en te onthouden	Declare omdat het hier niet altijd duidelijk is was de mogelijke flows zijn van de activiteiten.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
Heel veel pijlen die terug verwezen naar het begin van het model.Veel herhalingen van keuzes vb create content.	Ik had moeite met de negatieve lijnen (wat niet op elkaar mag volgen).	Duidelijker welke taken op welk moment uitgevoerd kunnen worden en wie ze begeleidt.	Declare omdat ik het vaak moeilijk vind om de verbanden te lezen. Vaak onduidelijk wat nu na elkaar mag en wat niet omdat het niet chronologisch is opgebouwd.
The circular paths going back to the start	Frequencies above activities and clearer succession rules	De geïntroduceerde symbolen zijn nog niet goed gekend. Beetje verwarrend dat activiteiten nu zowel de afgeronde vierhoeken kunnen zijn als ook op een lijn geplaatst kunnen worden.	BPMN en Declare ongeveer evenveel. BPMN was moeilijk te begrijpen door de complexiteit, Declare omdat ik de modelleringstaal niet meer goed beheers.
The BPMN model does not fully illustrate 'negative events' or in other words events that should not or cannot happen because of other limitations or constraints.	The DECLARE model does not provide additional information about objects when compared to the BPMN model that has 'notes'. On the other hand, it does clearly state which actions are not possible and is in that aspect easier to understand when compared to the BPMN model.	The BPMND + R model makes it easier to comprehend the whole process because it includes information about resources. It can also state which activities cannot be performed by the same resource, thus addressing 'negative events'.	DECLARE was the most difficult to understand because it was not clear which resources where responsible for which action.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
The many exclusive gateways made it diffucult to understand. Also, there were a lot of loops. What was easy is the fact that there are not many distinct activities.	The interpretation of the different symbols is confusing as there only are small differences.	The arrows are more clear than the arrows used in Declare.	Declare. There are only subtile differences in the meanings of the arrows. Oftentimes, this is confusing
Er stond veel duidelijke uitleg bijVaak een wir war van pijlen	Ik vond het moeilijk te begrijpen want mijn kennis over DECLARE is echt zeer beperkt.	Het is een nieuwe taal, dus nog moeilijk een beetje moeilijk om te begrijpen! De uitleg was wel duidelijk, enkel het toepassen was die altijd even duidelijk.	Declare omdat de constraints niet altijd even duidelijk zijn (en veel soorten)

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
Veel keuzes maken het verwarrend.De informatie artefacten zijn niet sluitend: ik kon niet alle vragen hierover correct beantwoorden.	Het is heel onduidelijk om alles in rekening te nemen tijdens het beantwoorden van de vragen. Alle activiteiten hebben veel verbindingen waardoor het zeer verwarrend is. Het model is te complex in declare.	Het gemakkelijke is dat deze terug in sequentie staat. Hierdoor is het iets duidelijker wat de voorziene volgorde is.Moeilijkheid is de opvolging van IN en EX door de BPMN gateways. Ik snap niet 100% wat wel en niet mag tussen seminarie 1 en 2. Het is duidelijk dat de defence hier niet tussen mag. Maar wat de rest betreft is die niet zo duidelijk. Zo lijk het feit dat de EX enkel defence bevat aan te geven dat voor de rest alles wel mag en zoveel als je wilt. Maar omdat de pijl wijst naar de gateway lijkt het alsof je op basis van het antwoord op de gateway nooit terug kan naar create content waardoor de rest ook niet meer gaat.	Declare was te complex voor dit proces. Er waren teveel beperkingen tussen alle activiteiten waardoor het heel moeilijk was om te zoeken wat wel en niet mag.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
Model was linear opgesteld, zonder complexe vertakkingen. Hierdoor was het model gemakkelijk te lezen en begrijpen.	De flow van het Declare model is minder duidelijk, er bestaan meer relaties dan in het BPMN model die het lezen en begrijpen moeilijker maakt. Eens de relaties begrepen zijn bakent het model wel duidelijker af.	De ex en in relaties stellen duidelijk wat kan en niet kan, zonder visueel complex te zijn (overvloed aan pijlen naar verschillende taken etc.). Daarnaast is het model linear opgebouwd wat de flow gemakkelijk begrijpbaar maakt.	DECLARE, het model bevat veel relaties. De relaties bakenen duidelijk af wat kan en niet kan. Echter ze dragen niet bij tot de leesbaarheid van het model. Hierdoor had ik meer tijd nodig om het model te begrijpen dan bij het BPMN EN BPRMND-R model.
Veel pijlen en gateways	Heel veel relaties, alles staat met elkaar in contact waardoor je heel veel relaties moet opzoeken om te kijken wat ze betekenen. Niet gebruiksvriendelijk	Al veel duidelijker, enkel vond ik het "niet gelijk aan" tekentje nog wat vager bij de supervisor.	Ik vond zowel BPMN als Declare niet gemakkelijk, er was veel te veel aan de hand. Het is ook al elven geleden dat ik er nog mee gewerkt heb, dus het waren zeer uitgebreide complexe modellen.
Once you understand what the process wants to achieve it is moderately easy to follow the different steps and gain insights or information.	In BPMN it's easier to follow the process, while DECLARE makes it a bit harder. The lack of extra information concerning the jury, supervisor... makes it hard to read. The connections between the different activities are harder to understand than with BPMN.	I found it easier to understand than DECLARE. More information is given.	DECLARE. A lack of information and the weird composition of the flow makes it harder to understand or gain insight. It is more compact but that does not outweigh the complexity.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
The large number of arrows makes the model more difficult to understand	To me, this model was more clear because I was able to consult the different relations while answering the questions.	For me, this modelling language is more comprehensible than the other two. The boxes with information about who's responsible for the task or who has to attend is much more clear than the use of messages in BMPN. Also, using the relation labels of declare, the use of many arrows is eliminated.	Voor mij is BPMN het moeilijkste om te begrijpen. Er wordt vaak gebruikt gemaakt van veel pijlen, wat voor mij het model ingewikkelder en minder overzichtelijk maakt. Ook vind ik het vaak moeilijk om te zien hoe vaak en wanneer een taak mag uitgevoerd worden. Dit is bij declare bijvoorbeeld al veel duidelijker. Het gebruik van bv precendence relations vind ik ook veel duidelijker dan het gebruik van gateways.

Continued on next page

Table I.1 – *Continued from previous page*

BPMN remarks	**Declare remarks**	**BPMND+R remarks**	**General remarks**
The BPMN model was easy enough to understand, as all tasks had a clear name and text annotations were provided when necessary. All XOR gateways were provided with a clear condition. Also, the model was relatively small. One thing that did make the model a little more difficult, was the number of loops (but I assume this is more due to the structure of the process, than due to the model).	What makes it more difficult, is that the various constraints are closely related to each other (e.g. does this constraint mean that A has to follow immediately after B, or that it just needs to follow some time, not necessarily right after B?). Also, there seem to be many different relationships, with symbols that look a lot alike. Nonetheless, I assume that I find the model mostly difficult because I'm not familiar enough with DECLARE. Perhaps if I had looked through more models, I would find the questions easier to tackle.	I found the model to be relatively easy to understand, as it has the same look and feel of a BPMN model: clear tasks, a clear sequential structure, gates with clear conditions. The new features (such as the number of times that a task can be performed and the conditions on resources) were also good extra touches. They were pretty straightforward, as I'm sure you could even deduce their meaning without any further explanations.	DECLARE. Firstly, because I'm not as familiar with DECLARE as I am with BPMN (I've already studied and made many BPMN models, so understanding this type of model and models related to it, comes more easily). Secondly, I also assume that even with more background to DECLARE, I'd still find it more complex, as there seem to be many types of closely related constraints, with symbols that look a lot alike and aren't quite straightforward.

Table I.1: Subjective feedback results