Say It In Your Own Words: Defining Declarative Process Models Using Speech Recognition*

Han van der Aa[1], Karl Johannes Balder[2], Fabrizio Maria Maggi[3], and Alexander Nolte[2,4]

[1] University of Mannheim, Germany
han@informatik.uni-mannheim.de
[2] University of Tartu, Estonia
Karl.Johannes.Balder@tudeng.ut.ee,alexander.nolte@ut.ee
[3] Free University of Bozen-Bolzano, Italy
maggi@inf.unibz.it
[4] Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Declarative, constraint-based approaches have been proposed to model loosely-structured business processes, mediating between support and flexibility. A notable example is the DECLARE framework, equipped with a graphical declarative language whose semantics can be characterized with several logic-based formalisms. Up to now, the main problem hampering the use of DECLARE constraints in practice has been the difficulty of modeling them: DECLARE's formal notation is difficult to understand for users without a background in temporal logic, whereas its graphical notation has been shown to be unintuitive. Therefore, in this work, we present and evaluate an analysis toolkit that aims at bypassing this issue by providing users with the possibility to model DECLARE constraints using their own way of expressing them. The toolkit contains a DECLARE modeler equipped with a speech recognition mechanism. It takes as input a vocal statement from the user and converts it into the closest (set of) DECLARE constraint(s). The constraints that can be modeled with the tool cover the entire Multi-Perspective extension of DECLARE (MP-DECLARE), which complements control-flow constraints with data and temporal perspectives. Although we focus on DECLARE, the work presented in this paper represents the first attempt to test the feasibility of speech recognition in business process modeling as a whole.

Keywords: Declarative Process Modeling, Speech Recognition, Natural Language Processing, Text Mining.

1 Introduction

Process models are an important means to capture information on organizational processes, serving as a basis for communication and often as the starting point for analysis and improvement [10]. For processes that are relatively structured, *imperative* process modeling notations, such as the Business Process Model and Notation (BPMN), are most commonly employed. However, other

* Work supported by the Estonian Research Council (project PRG887).

processes, in particular knowledge-intensive ones, are more flexible and, therefore, less structured. An important characteristic of such processes is that it is typically infeasible to specify the entire spectrum of allowed execution orders in advance [9], which severely limits the applicability of the imperative process modeling paradigm. Instead, such processes are better captured using *declarative* process models defined in process modeling languages, like DECLARE, whose semantics can be characterized using temporal logic properties. These models do not require an explicit definition of all allowed execution orders, but rather use constraints to define the boundaries of the permissible process behavior [5].

Although their benefits are apparent, establishing declarative process models is known to be difficult, especially for domain experts that generally lack expertise in temporal logics, and in most of the cases find the graphical notation of DECLARE constraints unintuitive [12]. Due to these barriers to declarative model creation, a preliminary approach has been presented in [1] that automatically extracts declarative process models from natural language texts, similar to others works that have investigated the generation of imperative process models from natural language descriptions (cf., [3, 11, 18, 22]).

In this work, we go beyond this state-of-the-art by presenting and evaluating an interactive approach, which takes vocal statements from the user as input, and employs speech recognition to convert them into the closest (set of) DECLARE constraint(s). The approach has been integrated into a declarative modeling and analysis toolkit. With this tool, the user is not required to have any experience in temporal logics nor to be familiar with the graphical notation of DECLARE constraints, but can express temporal properties using her/his own words. The temporal properties that can be modeled with the tool cover the entire Multi-Perspective Declare (MP-DECLARE) language that cannot only express control-flow properties, but also conditions on the data, time, and resource perspectives. A further important contribution of the paper is that the user evaluation presented in this paper can be considered the first test of the usability of speech recognition in business process modeling. This evaluation revealed that participants found that the tool would improve their efficiency, particularly in mobile settings. However, it also pointed towards a learning curve, especially for less experienced users. Based on the obtained feedback, we were able to make various improvements to the user interface.

The remainder of the paper is structured as follows. Section 2 introduces MP-DECLARE and the transformation of natural language into DECLARE. Section 3 presents our conceptual contributions to the generation of multi-perspective constraints based on natural language input, while Sect. 4 presents the implemented toolkit. Section 5 discusses the evaluation conducted to assess the usability of our tool. Finally, Sect. 6 considers related work, before concluding in Sect. 7.

2 Background

This section introduces MP-DECLARE followed by the main challenges associated with the translation of natural language into declarative constraints.

2.1 Declarative Process Modeling

DECLARE is a declarative process modeling language originally introduced by Pesic and van der Aalst in [19]. Instead of explicitly specifying the flow of the interactions among process activities, DECLARE describes a set of constraints that must be satisfied throughout the process execution. The possible orderings of activities are implicitly specified by constraints and anything that does not violate them is possible during execution. MP-DECLARE is the Multi-Perspective extension of DECLARE that was first introduced in [8] and can express constraints over perspectives of a process like data, time, and resources.

To explain the semantics of DECLARE and MP-DECLARE, we have to introduce some preliminary notions. In particular, we call a *case* an ordered sequence of events representing a single "run" of a process (often referred to as a *trace* of events). Each event in a trace refers to an *activity*, has a *timestamp* indicating when the event occurred, and can have additional *data attributes* as payload. Consider, e.g., the occurrence of an event *ship order* (O) and suppose that, after the occurrence of O at timestamp τ_O, the attributes *customer type* and *amount* have values *gold* and 155€. In this case, we say that, when O occurs, two special relations are valid *event*(O) and p_O(*gold*,155€). In the following, we identify *event*(O) with the event itself O and we call (*gold*,155€), the *payload* of O.

Template	**LTL semantics**	**Activation**
Participation(A)	$\mathbf{F}A$	A
Init(A)	A	A
Absence(A)	$\neg\mathbf{F}A$	A
AtMostOne(A)	$\neg\mathbf{F}(A \wedge \mathbf{X}(\mathbf{F}A))$	A
Responded Existence(A,B)	$\mathbf{F}A \rightarrow \mathbf{F}B$	A
Coexistence(A,B)	$\mathbf{F}A \leftrightarrow \mathbf{F}B$	A, B
Response(A,B)	$\mathbf{G}(A \rightarrow \mathbf{F}B)$	A
Chain Response(A,B)	$\mathbf{G}(A \rightarrow \mathbf{X}B)$	A
Precedence(A,B)	$\mathbf{G}(B \rightarrow \mathbf{O}A)$	B
Chain Precedence(A,B)	$\mathbf{G}(B \rightarrow \mathbf{Y}A)$	B
Not Coexistence(A,B)	$\mathbf{F}A \rightarrow \neg\mathbf{F}B$	A, B
Not Succession(A,B)	$\mathbf{G}(A \rightarrow \neg\mathbf{F}B)$	A, B

Table 1: Semantics for Declare templates

Declare A DECLARE model consists of a set of constraints applied to activities. Constraints, in turn, are based on templates. Templates are patterns that define parameterized classes of properties, and constraints are their concrete instantiations (we indicate template parameters with capital letters and concrete activities in their instantiations with lower case letters). Templates have a graphical representation and their semantics can be formalized using different logics [16], the main one being LTL over finite traces, making them verifiable and executable. Each constraint inherits the graphical representation and semantics from its template. Table 1 summarizes some Declare templates (the reader can refer to [5] for a full description of the language). Here, the **F**, **X**, **G**, and **U** LTL (future) operators have the following intuitive meaning: formula $\mathbf{F}\phi_1$ means that

ϕ_1 holds sometime in the future, $\mathbf{X}\phi_1$ means that ϕ_1 holds in the next position, $\mathbf{G}\phi_1$ says that ϕ_1 holds forever in the future, and, lastly, $\phi_1\mathbf{U}\phi_2$ means that sometime in the future ϕ_2 will hold and until that moment ϕ_1 holds (with ϕ_1 and ϕ_2 LTL formulas). The **O**, **Y**, and **S** LTL (past) operators have the following meaning: $\mathbf{O}\phi_1$ means that ϕ_1 holds sometime in the past, $\mathbf{Y}\phi_1$ means that ϕ_1 holds in the previous position, and, lastly, $\phi_1\mathbf{S}\phi_2$ means that sometime in the past ϕ_2 holds and since that moment ϕ_1 holds.

Consider, for example, constraint Response(a,b). This constraint indicates that if *a occurs*, *b* must eventually *follow*. Therefore, this constraint is satisfied for traces such as $\mathbf{t}_1 = \langle a, a, b, c\rangle$, $\mathbf{t}_2 = \langle b, b, c, d\rangle$, and $\mathbf{t}_3 = \langle a, b, c, b\rangle$, but not for $\mathbf{t}_4 = \langle a, b, a, c\rangle$ because, in this case, the second instance of a is not followed by a b. Note that, in $\mathbf{t}_2$, the considered response constraint is satisfied in a trivial way because a never occurs. An *activation* of a constraint in a trace is an event whose occurrence imposes, because of that constraint, some obligations on other events (targets) in the same trace. For example, a is an activation for Response(a,b) and b is a target because the execution of a forces b to be executed, eventually. In Table 1, for each template, the corresponding activations are specified.

Multi-Perspective Declare MP-Declare extends Declare with additional perspectives. We here describe its semantics informally and refer the interested reader to [8] for more details.

The standard semantics of Declare is extended by requiring additional conditions on data, i.e., the *activation condition*, the *correlation condition*, and a *time condition.* As an example, we consider constraint Response(ship order, send invoice), with *ship order* as activation and *send invoice* as target. The activation condition φ_a is a relation that must be valid when the activation occurs. If the activation condition does not hold the constraint is not activated. The activation condition has the form $p_A(x) \wedge r_a(x)$, meaning that when A occurs with payload x, the relation r_a over x must hold. For example, we can say that whenever *ship order* occurs, the order amount is higher than 100€, and the customer is of type *gold*, eventually an invoice must be sent. In case *ship order* occurs, but these conditions are not satisfied, the constraint is not activated.

The correlation condition φ_c is a relation that must be valid when the target occurs. It has the form $p_B(y) \wedge r_c(x, y)$, meaning that when B occurs with payload y, the relation r_c involving the payload x of A and the payload y of B must hold. For example, we can say that whenever *ship order* occurs with order amount higher than 100€, and customer type *gold*, eventually an invoice must be sent with the same order amount. Finally, a time condition can be specified through an interval ($I = [\tau_0, \tau_1)$) indicating the minimum and the maximum temporal distance allowed between the occurrence of the activation and the occurrence of the corresponding target.

2.2 From Natural Language to Declarative Models

A crucial component of our work involves the extraction of declarative constraints from natural language. This extraction step involves the identification

of the described actions (activities), as well as the identification of the constraint that applies to these actions. Due to the inherent flexibility of natural language, this extraction step can be highly challenging. Its difficulty manifests itself in the sense that, on the one hand, the same declarative constraint can be expressed in a wide variety of manners, whereas, on the other hand, subtle textual differences can completely change the meaning of the described constraint.

ID	Description
s_1	An invoice must be created before the invoice can be approved.
s_2	A bill shall be created prior to it being approved.
s_3	Invoice creation must precede its approval.
s_4	Approval of an invoice must be preceded by its creation.
s_5	Before an invoice is approved, it must be created.

Table 2: Different descriptions of Precedence(create invoice, approve invoice)

Variability of textual descriptions. As shown in Table 2, the same declarative constraint can be described in a broad range of manners. Key types of differences occur due to: the use of synonyms (e.g., *create invoice* in s_1 and *create bill* in s_2) and due to different grammatical structures (e.g., s_1 uses verbs to denote activities, whereas s_3 uses nouns, like "*invoice creation*"). Finally, constraint descriptions can differ in the order in which they describe the different components of binary constraints, i.e., whether they describe the constraint in a chronological fashion, e.g., s_1 to s_3, or in the reverse order, such as s_4 and s_5. To support users in the elicitation of declarative process models, an approach must not limit users too much in terms of the input that they can provide. Rather, an approach must accommodate different manners in which users may describe constraints. However, a successful approach must be able to do this while also being able to recognize subtle distinctions among constraint types.

Subtle differences leading to different constraints. Small textual differences can have a considerable impact on the semantics of constraint descriptions and, thus, on the constraints that should be extracted from them. To illustrate this, consider the descriptions in Table 3. In comparison to description s_6, the three other descriptions each differ by only a single word. However, as shown in the right-hand column, the described constraints vary greatly. For instance, the difference between the Response constraint of s_6 and the Precedence constraint described by s_7 lies in the obligation associated with the *send invoice* action. The former specifies that this *must* occur, whereas the latter specifies that it *can* occur. Further, the direction in which a constraint is described is often signaled through small textual elements, typically through the use of temporal prepositions. For instance, in s_8, the use of *first* completely reverses the meaning of the described constraint. Finally, the presence of a negation also drastically changes

the meaning of a constraint, as seen for s_9. The addition of *not* to description s_6 changes Response into Not Succession.

ID	Description	Constraint
s_6	If an order is shipped, an invoice must be sent.	Response(A,B)
s_7	If an order is shipped, an invoice can be sent.	Precedence(A,B)
s_8	If an order is shipped, an invoice must be sent first.	Precedence(B,A)
s_9	If an order is shipped, an invoice must not be sent.	Not Succ.(A,B)

Table 3: Subtle textual differences (A as *ship order*, B as *send invoice*)

State of the art. So far, only one approach has been developed to automatically extract declarative constraints from natural language text. This approach, by van der Aa et al. [1], is able to extract five types of Declare templates, Init, End, Precedence, Response, Succession, as well as their negated forms. Its evaluation results show that it is able to handle a reasonable variety of inputs. Recognizing its potential as well as its limitations, we extend this approach as follows: (1) we generalize the pattern recognition mechanisms in order to handle more flexible inputs, (2) we cover eight additional constraint templates, and (3) we add support for augmentation with data and time conditions.

3 Conceptual Approach

Fig. 1 provides an overview of the main components of our Speech2RuM approach. As shown, the user provides inputs through speech as well as the interaction with the Graphical User Interface (GUI). In this way, users are able to construct a declarative process model through three main functions: (1) using speech to describe constraints in natural language, (2) augmenting constraints with data and time conditions, and (3) editing and connecting the constraints.

In this section, we outline our conceptual contributions with respect to the first two functions. We cover the implementation of the approach in Sect. 4. There, we also show how a model can be edited after its creation.

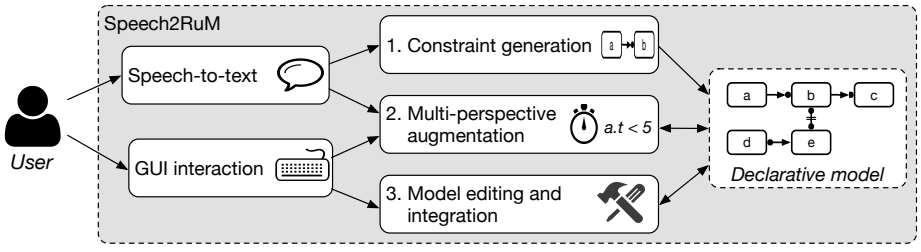


Fig. 1: Overview of the Speech2RuM approach

3.1 Constraint generation

In this component, our approach turns a constraint description, recorded using speech recognition, into one or more constraints. For this task, we take the result of the parsing step of the state-of-the-art approach [1] as a starting point. Given a sentence S, parsing yields a list A_S of actions described in the sentence and the interrelations that exist between the actions, i.e., a mapping $rel_S : A_S \times A_S \rightarrow relationType$, with $relationType \in \{xor, and, dep\}$. As depicted in Fig. 2, an action $a \in A_S$ consists of a verb and optional subjects and objects.

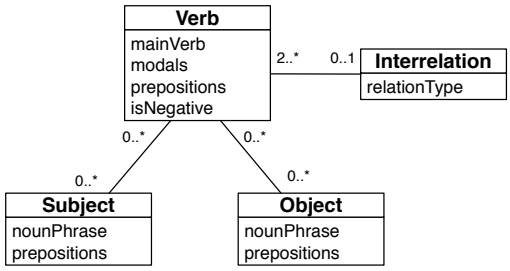


Fig. 2: Semantic components returned in the parsing step of [1]

Based on this parsing step, we have added support to handle eight additional constraint types, presented in Table 4. Given a set of activities A_S and a relation rel_S extracted for a sentence S, these types are identified as follows:

- **Participation and Absence.** If A_S contains only one action, either a Participation or an Absence constraint is established for the action, depending on whether it is negative or not.
- **AtMostOne.** If a Participation constraint is originally recognized, we subsequently check if S specifies a bound on its number of executions, i.e., by checking for phrases such as *at most once*, *not more than once*, *one time*.

Constraint	**Example**
Participation	*An invoice must be created*
Absence	*Dogs are not allowed in the restaurant.*
AtMostOne	*An invoice should be paid at most once.*
Coexistence	*An order should be shipped and paid.*
Responded Existence	*If a product is produced, it must be tested.*
Not Coexistence	*If an application is accepted, it cannot be rejected.*
Chain Precedence	*After an order is received, it may immediately be refused.*
Chain Response	*After an order is received, it must directly be checked.*

Table 4: Additional constraint types in Speech2RuM

- **Coexistence.** Coexistence relations are identified for sentences with two actions in an *and* relation, typically extracted from a coordinating conjunction like *shipped and paid*. Furthermore, some notion of obligation should be present, in order to distinguish between actions that *can* happen together and ones that *should*.
- **Responded Existence.** Responded Existence constraints are extracted when two actions are in a dependency relation, i.e., a_1 *dep* a_2, for which it holds that the target action, a_2, is indicated to be mandatory, e.g., using *must*. The key difference between Response and Responded Existence is that the former includes a notion of order, i.e., a_1 precedes a_2, whereas no such order is specified for Responded Existence constraints.
- **Not Coexistence.** This constraint type is identified as the negated form of both Coexistence and Responded Existence. Semantically, in both cases, the description states that two actions should not appear in the same process instance. This is, e.g., seen in Table 4, where "*If an application is accepted, it cannot be rejected*" is actually the negative form of a Responded Existence description.
- **Chain Precedence and Chain Response.** These constraint types are specializations of Precedence and Response constraints. Chain constraints are recognized by the presence of a temporal preposition that indicates immediacy. Generally, such a preposition is associated with the verb of either action a_1 or a_2 in a relation a_1 *dep* a_2. For this, we consider the preposition *immediately* and several of its synonyms, i.e., *instantly*, *directly*, and *promptly*.

3.2 Multi-Perspective Augmentation

Our approach supports the augmentation of constraints with conditions on the data and time perspectives, turning DECLARE constraints into MP-DECLARE ones. Our approach allows users to express three types of conditions through speech recognition, i.e., *activation*, *correlation*, and *time* conditions (see Sect. 2.1).

We note that descriptions of conditions likely reflect textual fragments rather than full sentences. Furthermore, given that these are short statements, the expected variance is considerably lower than for descriptions of declarative constraints. For these reasons, our approach to extract conditions is based on pattern matching as opposed to the grammatical parsing used in Sect. 3.1.

Activation conditions. Activation conditions denote specific requirements that must be met in order for a constraint to be applicable, e.g., stating that Response(ship order, send invoice) should only apply when the amount associated with *ship order* activity is above 500. Our approach allows users to express complex conditions, i.e., conditions that concatenate multiple logical statements, such as "*The amount is higher than 500 and the color is not red*". Therefore, given an input string S, our approach first splits S into sub-strings, denoted by $split(S)$. This is done by recognizing the presence of coordinating conjunctions (*and* and *or*) and dividing S accordingly. Each sub-string $s \in split(S)$ is

Condition	Pattern	Example
$>$ or $\geq$	[greater|higher|more] than [or equal to]	*Amount higher than 500.*
$<$ or $\leq$	[smaller|lower|less] than [or equal to]	*Quantity is less than or equal to 12.*
$=$ or $\neq$	is [not] [equal to]	*The color is not red.*
$\in$ or $\notin$	is [not] in [*list*]	*Size is not in small, medium, large.*

Table 5: Supported patterns for activation conditions

expected to correspond to an individual expression, which together are joined using logical $\wedge$ and $\vee$ operators. On each $s \in split(S)$, we use pattern matching based on the patterns depicted in Table 5. In this manner, we are able to handle conditions related to both numerical, (e.g., *amount* and *length*) and categorical attributes (e.g., *color* and *customer type*).

Correlation conditions. Correlation conditions must be valid when the *target* of a constraint is executed. These express relations that must exist between attributes associated with the activation and the target of the constraint, e.g., the employee *receiving* a loan application should not be the same as the employee that *checks* it. Our approach handles two patterns here, either allowing a user to express that an attribute should be equal for both activities, e.g., "*The amount is the same*" or that they not equal, e.g., "*The applicant is different*".

Condition	Pattern	Example
$time \leq x$	[in|at most|no later than] [x]	*At most 3 hours*
$x \leq time \leq y$	between [x] and [y]	*Between 3 and 5 days*
$x \leq time \leq y$	not [before|earlier than] [x] and [within|not later than|not after] [y]	*Not before 3 hours and within 12 hours*

Table 6: Supported patterns for time conditions

Time conditions. A time condition bounds the time between the occurrence of the activation and target of a constraint, e.g., stating that the target of a Response constraint should occur within 5 days after its activation. As shown in Table 6, we allow users to specify time conditions according to three general patterns. The first pattern only specifies an upper bound on the duration, whereas the other two patterns specify ranges. Note that we support time conditions specified using seconds, minutes, hours, and days.

4 The Tool

We integrated the proposed Speech2RuM approach into RuM, a modeling and analysis toolkit for Rule Mining.[1] The tool is implemented in Java 11 and uses the

[1] The tool can be found at `https://sep.cs.ut.ee/Main/RuM`

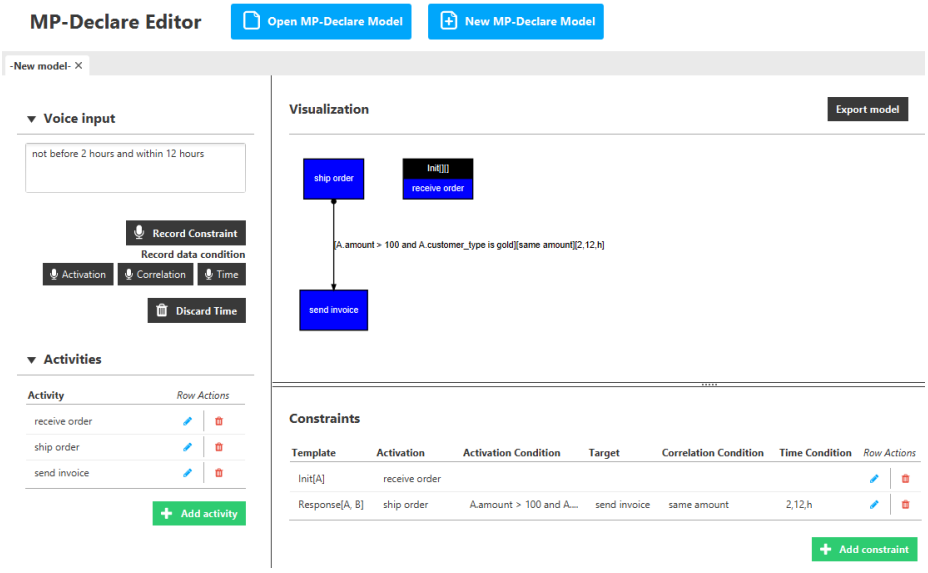
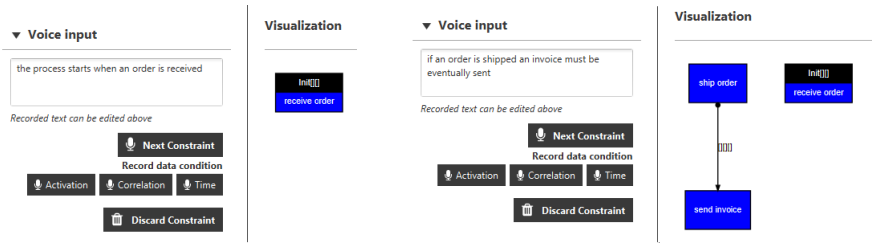


Fig. 3: Screenshot of the Speech2RuM implementation



(a) Inserting an Init constraint (b) Inserting a Response constraint

Fig. 4: Inserting control-flow properties

speech recognition API provided by *Google Cloud Speech*.[2] In Fig. 3, a screenshot of the latest version of the tool is shown. In the top-left area of the screen, the recognized vocal input is shown. It is possible to record a constraint (control-flow perspective) as shown in Fig. 4 where an Init and a Response constraint are modeled. After having inserted a DECLARE constraint, the user can record an activation, a correlation, and a time condition as shown in Fig. 3 where an activation (*amount* is greater than 100 and *customer type* is *gold*) and a correlation (activation and target share the same value for attribute *amount*) condition have already been recorded and a time condition is extracted from the vocal input *"not before 2 hours and within 12 hours"*. The constraints expressed

[2] `https://cloud.google.com/speech-to-text/`

with their graphical notation are shown in the top-right area of the screen, and modifiable lists of activities and constraints are shown in the bottom part of the screen.

5 User Evaluation

To improve the current implementation and to assess the feasibility of the developed Speech2RuM approach based on user feedback, we conducted a qualitative user study. Our aim was to (1) identify means for improving the interface, (2) identify issues and demands of individuals with different backgrounds, and (3) identify usage scenarios. In the following, we will describe the study before discussing our findings and potential threats to validity.

5.1 Study

Participants. For our study, we selected eight participants (P1 to P8) with differing characteristics. We selected two participants that had experience related to Business Process Management (BPM, tier 1, P6 and P7), two participants with formal background on temporal logics (tier 2, P3 and P8), two participants that had used DECLARE to model temporal constraints (tier 3, P2 and P4), and two participants that had used different tools to model and analyze temporal constraints with DECLARE (tier 4, P1 and P5). We chose this differentiation to study how individuals with different levels of expertise perceive the tool and identify potential issues and demands.

Study setup. The study was conducted via Skype by a team consisting of a *facilitator* and an *observer*, with the facilitator guiding the participant and the observer serving in a supporting role. We used a manufacturing process in a bicycle factory[3] to provide a scenario that is likely to be familiar for all participants. We opted to use a common scenario rather than asking participants to describe a process of their choice to ensure the comparability of our findings. We also created a help document[4] to outline the tool's capabilities to the participants.

Prior to the study, the facilitator sent the participants a link to the scenario, the help document, and the tool itself suggesting to read the documents and install the tool. At the beginning of the study, the facilitator introduced the study procedure and the tool. When ready, the participant shared her/his screen, whereas the facilitator started the video recording and began guiding the participant through the prepared scenario. The scenario required the participants to construct a declarative process model. First, they added constraints by reading prepared sentences, before being asked to establish their own constraint descriptions. During this time, the facilitator and the observer noted down any issues the participant mentioned or that appeared to come up.

[3] The scenario can be found at `https://git.io/JfHbl`
[4] The help document can be found at `https://git.io/JfHbc`

At the end of the study, the facilitator conducted a follow-up interview, focusing on clarifying questions about issues the participant had encountered. He also asked the participants what s/he *"liked about the tool"*, *"wished would be different"*, and *"under which circumstances s/he would use it"*. After the study, the participants were asked to complete a short post-questionnaire. The questionnaires consisted of multi-point Likert scales including the System Usability Scale (SUS) [7] and scales covering satisfaction, expectation confirmation, continuation intention, and usefulness which were adapted from the ones presented by Bhattacherjee in [6] and used to assess continued use of information systems.[5] The individual studies lasted between 14 and 35 minutes each.

Result analysis. To analyze the collected data, the research team first built an affinity diagram [13] based on the video recordings, observations, and follow-up interviews, by creating a single paper note for each issue that was reported. The team then clustered the notes based on emerging themes, which resulted in 139 notes divided over 21 distinct clusters. The results of the questionnaires served as an additional qualitative data point during the analysis.

5.2 Findings

In general, our study revealed that the tested interface was reasonably usable as evidenced by an average SUS score of 73.13, which can be considered to be a good result compared to the commonly reported average of 68 [7]. However, our study also revealed usability issues, which served as a basis to improve the tool (Fig. 3). The main changes include the possibility to undo changes to the model based on the last recording, editing the recorded text directly rather than having to re-record it, and adding data conditions through speech recognition (instead of specifying those conditions manually).

Common observations. First, we found that all participants **wrote sentences down** before saying them aloud. They either wrote them *"on paper"* (P1), *"typed them on [their] computer"* (P2), or noted down key parts they wanted to say (P7). Most participants also used somewhat **unnatural sentences** when entering constraints. This is evidenced by them, e.g., using activity names instead of natural sentences (*"after [activity1], [activity2] is executed."*, P7). Some even tried to identify keywords that the algorithm might pick up (*"is activity like a keyword?"*, P8). Only P1 used natural sentences from the start. Taking these findings together indicates that users might have felt insecure to just use natural sentences and rather tried to create sentences that they felt the system would be able to understand, but that were difficult for them to formulate without writing them down first.

Trying to identify keywords and adapting the input statements for the tool might have **increased the complexity of entering constraints** while being counterproductive when using speech recognition since our tool was designed to understand natural expressions. These issues indicate that **individuals need to**

[5] The complete questionnaire can be found at `https://git.io/JfHb8`

learn how to effectively use speech as a means of modeling constraints regardless of their previous experience related to BPM or DECLARE. Therefore, it would be desirable for users to go through a training phase to better understand how to interact with the tool and also appreciate the variety of vocal inputs the tool can deal with.

Difference between participants with different backgrounds. There were also noticeable differences between individuals from different backgrounds. For example, participants from tiers 1 and 2 (having low confidence with DECLARE) **mainly relied on the visual process model** to assess whether the tool had understood them correctly, while participants from tiers 3 and 4 (more familiar with DECLARE and analysis tools based on DECLARE) **mainly relied on the list providing a semistructured representation** of the entered constraints (bottom-right area in Fig. 3). This was evidenced by how they tried to fix potential errors. Participants from tiers 1 and 2 initially tried to edit the visual process model (observation of P6, *"the activity label should be editable"*, P2) – which was not possible in the version of the tool we tested – while participants from tiers 3 and 4 directly started editing the constraint list (observation of P1 and P5). Only P2 attempted both.

Findings from our study also indicated that the way the **recognized text** was displayed (text field below *Voice input* in the top-left area of the screenshot in Fig. 3) might not have been ideal. Participants checked the input correctness after they entered the first constraint, but did not continue to do so afterward (observation of P3 and P6) despite few sentences being wrongly translated from speech into text during the tests. This led to confusion, particularly among less experienced participants, who thought that they did not formulate a constraint correctly, while the issue was that their vocal input was not recorded properly. One way to address this issue is to highlight the text field containing the recognized text directly after a new text has been recorded to indicate to the user that s/he should pay attention to the recognized text.

These findings indicate that, even if using speech recognition users can somehow bypass the interaction with the graphical notation of DECLARE constraints, **it still appears difficult for less experienced users to assess the correctness of their entered constraints**. One approach to mitigate this issue might be to provide the option to mark constraints that a user cannot verify and subsequently ask an experienced user to assess their correctness. In alternative, traces representing examples and counterexamples of the recorded behaviors together with a textual description of the constraint could be shown.

The aforementioned issues might also have led to **less experienced participants perceiving the tool as less useful** ($m = 2.63$ for tier 1 and $m = 3.50$ for tier 4) and being **less inclined to use the tool in the future** ($m = 2.33$ for tier 1 and $m = 3.65$ for tier 4) than experienced participants. Related to this finding, it thus appears counter-intuitive that experienced participants were slightly less satisfied with the tool than less experienced participants ($m = 3.00$ for tier 3, $m = 3.50$ for tier 4, and $m = 3.67$ for tiers 1 and 2). This can however potentially be explained by experienced participants having higher expectations

regarding the functionality of the tool (*"what about recording data conditions as well?"*, P5).

Finally, it should be noted that participants from different backgrounds were *"excited"* (P5) to use the tool. Some even continued to record constraints after the actual evaluation was over (*"let me see what happens when I [...]"*, P8). It thus appears reasonable to assume that addressing the identified shortcomings can positively influence the perception of the tool.

Potential usage scenarios. The participants mentioned multiple scenarios during which they would consider speech as useful for entering constraints. Most of them stated that using speech would **improve their efficiency** because they could *"quickly input stuff"* (P4) especially when reading *"from a textual description"* (P1) with P5 pointing out that it would be particularly useful for *"people that are not familiar with the editor"* (P5). On the other hand, P8 had a different opinion stating that *"manually is quicker"* (P8). S/he did however acknowledge that speech might be useful *"when I forget the constraint name"* (P8). Moreover, participants also suggested that vocal input might be useful *"for designing"* (P3) which points towards its usefulness for specific activities at the early stages of modeling a process. Finally, P1 mentioned that speech input would be useful for **mobile scenarios** such as *"using a tablet"* (P1), because entering text in a scenario like that is usually much more time consuming than when using a mechanical keyboard.

5.3 Threats to validity

The goal of our study was to collect feedback for improving the tool, identify potential usage scenarios, and pinpoint any issues for individuals with different backgrounds. It thus appeared reasonable to conduct an in-depth qualitative study with selected participants from a diverse range of backgrounds related to their knowledge and experience with BPM in general and temporal properties and DECLARE in particular. Conducting a study with a small sample of participants is common because research has shown that the number of additional insights gained deteriorates drastically per participant [17]. There are, however, some threats to validity associated with this particular study design. A first threat is related to the fact that we developed a specific tool and studied its use by specific people in a specific setting over a limited period of time. Despite carefully selecting participants and creating a setting that would be close to how we envision the tool would be commonly used, it is not possible to generalize our findings beyond our study context since conducting the same study with different participants using a different setup might yield different results. In addition, the study results were synthesized by a specific team of researchers which poses a threat to validity since different researchers might potentially interpret findings differently. We attempted to mitigate this threat by ensuring that observations, interviews, and the analysis of the obtained data were collaboratively conducted by two researchers. We also abstained from making causal claims providing instead a rich description of the behavior and reporting perceptions of participants.

6 Related work

Organizations recognize the benefit of using textual documents to capture process specifications [4, 23], given that these can be created and understood by virtually everyone [11]. To allow these documents to be used for automated process analysis, such as conformance checking, a variety of techniques have been developed to extract process models from texts [11, 18, 22]. Other works exploit textual process specifications for model verification [2, 20] or directly for process analysis [3, 21]. In the context of declarative process models, some recent works also provide support for the extraction of DCR graphs from textual descriptions [14, 15], whereas the preliminary work on the extraction of DECLARE constraints from texts presented in [1] represents the foundation of our work.

7 Conclusion

In this work, we presented an interactive approach that takes vocal statements from the user as input and employs speech recognition to convert them into multi-perspective, declarative process models. Our Speech2RuM approach goes beyond the state-of-the-art in text-to-constraint transformation by covering a broader range of DECLARE templates and supporting their augmentation with data and time conditions. The integration of our approach into the RuM toolkit enables users to visualize and edit the obtained models in a GUI. Furthermore, it also allows these models to directly serve as a basis for the toolkit's analysis techniques, such as conformance checking and log generation. Finally, we note that the conducted user evaluation represents the first study into the feasibility of using speech recognition for business process modeling. The results demonstrated its promising nature, although modeling purely based on speech recognition is especially suitable for mobile environments.

In future work, we aim at further developing Speech2RuM based on the obtained user feedback. In particular, we are going to integrate the speech recognition tool with a chatbot which represents a valid alternative in the context of desktop applications. In addition, the text-to-constraint component shall be improved to support less natural descriptions, e.g., those that explicitly mention the term *activity* to denote a process step. Finally, it will be highly interesting to investigate how speech recognition can be lifted to also support the elicitation of imperative process models.

References

1. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: CAiSE. pp. 365–382. Springer (2019)
2. van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models: The automatic detection of inconsistencies. Inf. Syst. 64, 447–460 (2017)
3. van der Aa, H., Leopold, H., Reijers, H.A.: Checking process compliance against natural language specifications using behavioral spaces. Inf. Syst. 78, 83–95 (2018)

4. van der Aa, H., Leopold, H., van de Weerd, I., Reijers, H.A.: Causes and consequences of fragmented process information: Insights from a case study. In: AMCIS (2017)
5. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. CSRD 23(2), 99–113 (2009)
6. Bhattacherjee, A.: Understanding information systems continuance: an expectation-confirmation model. MIS quarterly pp. 351–370 (2001)
7. Brooke, J., et al.: SUS-a quick and dirty usability scale. Usability evaluation in industry 189(194), 4–7 (1996)
8. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. 65, 194–211 (2016)
9. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. JoDS 4(1), 29–57 (2015)
10. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)
11. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. pp. 482–496. Springer (2011)
12. 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)
13. Holtzblatt, K., Wendell, J.B., Wood, S.: Rapid contextual design: a how-to guide to key techniques for user-centered design. Elsevier (2004)
14. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: From texts to declarative processes and back. BPM (Demos) 2196, 66–70 (2018)
15. López, H.A., Marquard, M., Muttenthaler, L., Strømsted, R.: Assisted declarative process creation from natural language descriptions. In: EDOCW. pp. 96–99. IEEE (2019)
16. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. ACM Transactions on the Web 4(1) (2010)
17. Nielsen, J., Landauer, T.: A mathematical model of the finding of usability problems. In: SIGCHI. pp. 206–213 (1993)
18. de Oliveira, J.P.M., Avila, D.T., dos Santos, R.I., Fantinato, M.: Assisting process modeling by identifying business process elements in natural language texts. In: ER Workshops, p. 154. Springer (2017)
19. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC. pp. 287–300 (2007)
20. Sànchez-Ferreres, J., van der Aa, H. and Carmona, J., Padró, L.: Aligning textual and model-based process descriptions. Data Knowl. Eng. 118, 25–40 (2018)
21. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L.: Formal reasoning on natural language descriptions of processes. In: BPM. pp. 86–101. Springer (2019)
22. Schumacher, P., Minor, M., Schulte-Zurhausen, E.: Extracting and enriching workflows from text. In: IRI. pp. 285–292. IEEE (2013)
23. Selway, M., Grossmann, G., Mayer, W., Stumptner, M.: Formalising natural language specifications using a cognitive linguistic/configuration based approach. Inf. Syst. 54, 191–208 (2015)