UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Karl Johannes Balder

Defining Business Process Models with Natural Language Processing and Speech Recognition

Bachelor's Thesis (9 ECTS)

Supervisor: Fabrizio Maria Maggi, PhD

Tartu 2020

Defining Business Process Models with Natural Language Processing and Speech Recognition

Abstract: Proposed approaches for modeling unstructured business processes include declarative, constraint-based solutions, which meet halfway between support and flexibility. A noteworthy example is the Declare framework, which is equipped with a graphical declarative language whose semantics can be expressed with multiple logic-based formalisms. So far, the practical use of Declare constraints has been mostly hampered by the difficulty of modeling them: the formal notation of Declare represents a steep learning curve for users lacking an understanding of temporal logic, while the graphical notation has proven to be unintuitive. As such, this work presents and assesses an analysis toolkit which tries to circumvent this issue by providing the user with a possibility to model Declare constraints by using their own way of expression.

The toolkit includes a Declare modeler supplied with a speech recognition mechanism, which accepts a user's vocal statement as input and converts it into the closest Declare constraint(s) by combining voice recognition, natural language processing and business rule extraction technologies. The constraints cover the entire Multi-Perspective extension of Declare (MP-Declare), complementing control-flow constraints with data and temporal perspectives. Even though this thesis focuses on Declare, it represents the first attempt at testing the practicability of speech recognition in business process modeling altogether.

Keywords: Business Process Management, Declarative Process Models, Process mining, Process discovery, Conformance checking, Process analytics tool

CERCS: P170 - Computer Science

Äriprotsessi mudelite määratlemine loomuliku keele töötluse ja kõnetuvastuse abil

Lühikokkuvõte: Poolstruktureeritud äriprotsesside modelleerimiseks on välja pakutud deklaratiivseid, kitsendustel põhinevaid lahendusi, mis tasakaalustavad tuge ja paindlikkust. Siinkohal on väljapaistev näide Declare raamistik, mis on varustatud graafilise deklaratiivse keelega, mille semantikat on võimalik väljendada mitme loogikal põhineva formalismiga. Siiani on Declare kitsenduste praktilist kasutust hoidnud tagasi nende modelleerimise raskus: Declare'i formaalne notatsioon loob kasutajate jaoks, kellel puuduvad teadmised ajalisest loogikast, järsu õppimiskõvera, samas kui graafiline notatsioon on osutunud vaistuvastaseks. Käesolev lahendus pakub välja ning hindab analüüsi-riistakomplekti, mille eesmärgiks on vältida mainitud probleemi, pakkudes kasutajale võimaluse modelleerida Declare kitsendusi, kasutades isiklikku väljendusviisi.

Väljapakutavasse riistakomplekti kuulub kõnetuvastuse mehhanismiga Declare modelleerija, mis võtab sisendina vastu kasutaja häälteate ning muundab selle lähima(te)ks Declare kitsendus(t)eks, ühendades hääletuvastuse, loomuliku keele töötluse ja ärireeglite tuletamise tehnoloogiad. Kitsendused katavad kogu Multi-Perspective laienduse Declare'ist (MP-Declare), toetades juhtimisvoo kitsendusi andmelise ning ajalise perspektiiviga. Kuigi käesolev töö keskendub Declare'ile, on see üleüldse esimene katsetus testida kõnetuvastuse teostatavust äriprotsesside modelleerimisel.

Võtmesõnad: Äriprotsesside haldus, deklaratiivsed protsessimudelid, protsessikaeve, deklaratiivse protsessikaeve mudelid, protsesside väljaselgitamine, vastavuskontroll, protsessianalüütika tööriist

CERCS: P170 - Arvutiteadus

Table of Contents

Introduction

Business processes are composed of activities performed by an organisation in order to accomplish a business goal [1]. In this way one could describe, for example, an order-to-pay process, i.e., a business process for receiving and processing customer orders, as a sequence of activities. Business Process Management (BPM) is the field of computer science studying the lifecycle of a business process from the design phase to the execution to the monitoring and improvement [1]. Business processes can be complex and unpredictable (a.k.a. unstructured processes) [2]. These types of processes are difficult to model with standard procedural process modeling languages like BPMN [2] (because these languages would produce complex "spaghetti-like" models that are impossible to understand), but are more suitable to be represented with declarative process modeling languages like Declare [2].

RuM [3] is a process mining [4] tool implementing several techniques for process analysis and improvement using information included in process execution logs and is based on the declarative process modeling language Declare [3]. The tool's advantage is in the interconnection of different declarative process mining techniques, which the user can combine. However, the tool requires previous knowledge of declarative process modeling languages in general and of Declare in particular, which is difficult to use in practice since its formal notation represents a steep learning curve for users lacking an understanding of temporal logic, while its graphical notation has proven to be unintuitive [5].

This project tries to solve this issue by adding a voice recognition functionality to the modeling component of RuM, which would allow the user to create Declare models from speech. This will give the user the ability to create declarative process models from natural language, without the need for previous knowledge about declarative process modeling languages. The constraints covered by the project include the entire Multi-Perspective extension of Declare (MP-Declare), complementing control-flow constraints with data and temporal perspectives.

This thesis is structured as follows. The first chapter will give an overview of business process modeling, Declare and text mining. The proposed approach is described in the second chapter with the implementation introduced in the third chapter. The fourth chapter will provide a user evaluation of the tool. The fifth chapter presents related work, while the sixth chapter concludes the thesis.

1. Background

This chapter introduces the background about business process models and its types which are related to the goals of the project. An overview of text mining is also given.

1.1. Types of process models

Process models are used to configure workflow management systems supporting the execution of business processes [2]. Procedural models like BPMN [2] represent step by step how the process is expected to be executed. On the other hand, declarative process models consist of constraints on activities and every behaviour that does not violate them is allowed [3].

Example 1: Declarative process model containing constraints on the control flow only.

Constraints:

1. Activity A appears at least once in the process execution.
2. Each time activity A appears in the process execution, activity B must immediately follow.
3. The process execution ends with activity B.

Declarative process modeling languages like Multi-Perspective Declare (MP-Declare) [6] allow the user to set constraints on data; one such constraint is presented in Example 2.

Example 2: The process execution does not include activity A with an “amount” attribute greater than 200 euros.

1.3. Declare

It is possible to express declarative process models with the Declare [2]. Declare is a declarative process modeling language based on parameterized temporal patterns (a.k.a. templates) that can be instantiated as constraints over activities constraining the number of occurrences of activities or their order. For a process execution (a.k.a. trace) to conform with a Declare model, it must satisfy all of the constraints set up in the model.

Standard Declare includes in total 19 different templates, of which 5 are existence templates, including just one parameter; 11 are relation templates, describing a link between two parameters; 3 are negative relation templates [7].

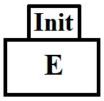
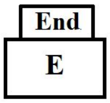
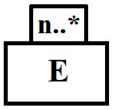
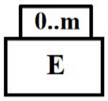
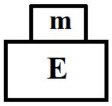
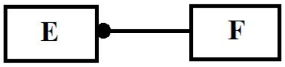
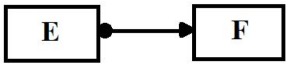
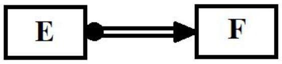
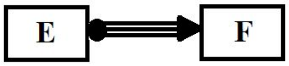
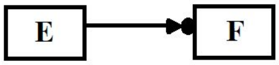
Name	Description	Representation
Init(E)	The trace must start with E	
End(E)	The trace must finish with E	
Existence(n, E)	E must occur in the trace at least n times	
Absence(m, E)	E must occur in the trace at most (m-1) times	
Exactly(m, E)	E must occur in trace exactly m times	

Table 2: Existence templates [3]

Name	Description	Representation
Responded Existence(E, F)	If E occurs, then F occurs as well	
Response(E, F)	If E occurs, then F occurs after E	
Alternate Response(E, F)	If E occurs, then F occurs afterwards before E recurs	
Chain Response(E, F)	If E occurs, then F occurs immediately after	
Precedence(E, F)	F can occur only if it is preceded by E	

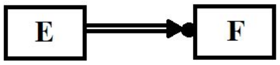
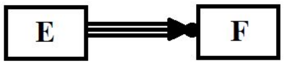
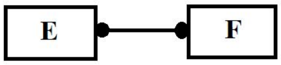
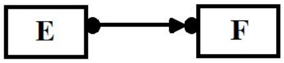
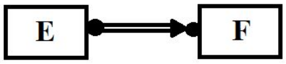
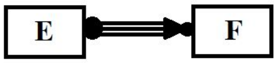
Alternate Precedence(E, F)	F can occur only if it is preceded by E and no other F recurs in between	
Chain Precedence(E, F)	F can occur only if it is immediately preceded by E	
Co-Existence(E, F)	If E occurs then F occurs and vice versa	
Succession(E, F)	F must occur after E and E must occur before F	
Alternate Succession(E, F)	F must occur after E and E must occur before F and they alternate each other	
Chain Succession(E, F)	F must occur immediately after E and E must occur immediately before F	

Table 3: Relation templates [3]

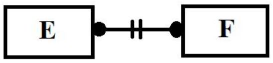
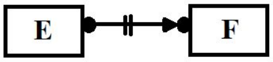
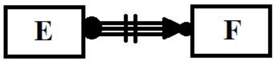
Name	Description	Representation
Not Co-Existence(E, F)	E and F never occur together	
Not Succession(E, F)	F never follows E	
Not Chain Succession(E, F)	F cannot occur immediately after E	

Table 4: Negative relation templates [3]

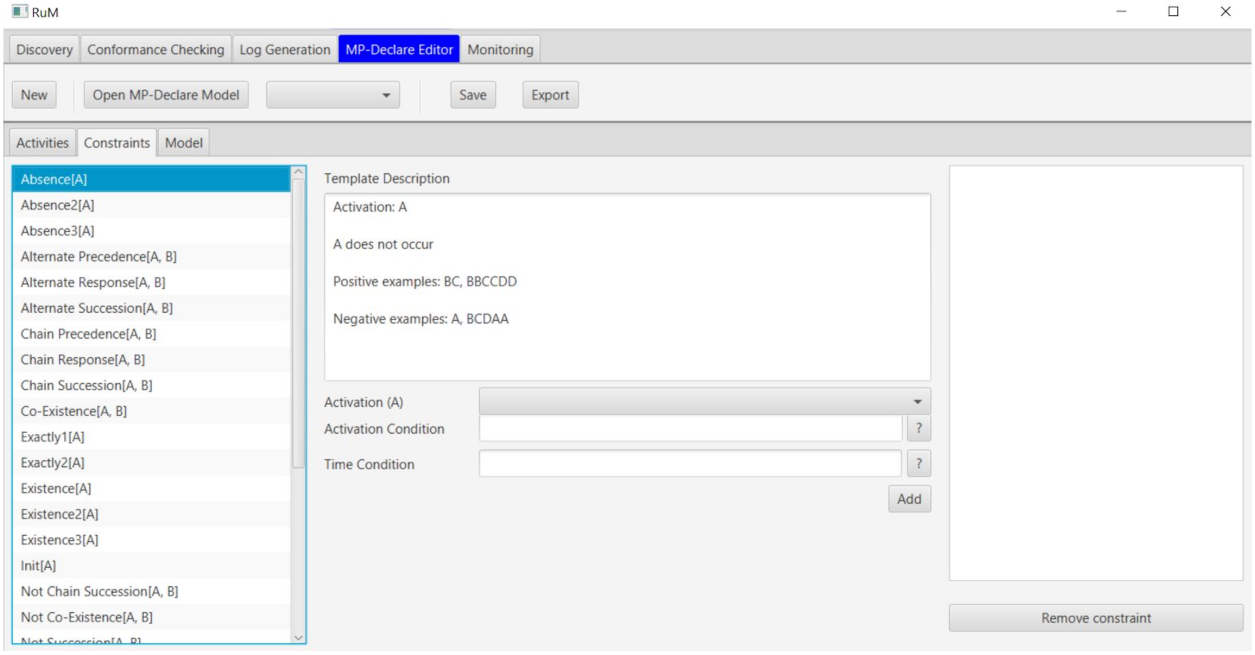
1.3.1. Multi-Perspective Declare

Multi-Perspective Declare (MP-Declare) extends Declare, allowing the user to set conditions on data in addition to control flow. In a process execution, when an activity occurs, it can be attached to a data payload, i.e., to a list of pairs attribute-values that represent how the activity has modified the value of those data attributes after its execution. In addition, each Declare constraint has an activation and a target. Intuitively, an application "triggers" a constraint in the sense that its occurrence influences the (non-)occurrence of another activity (the target of the constraint). Possible data conditions are, therefore, activation conditions constraining the attribute values of the payloads of the activations of a constraint; correlation conditions constraining the attribute values of the payloads of the targets of a constraint or correlating the data payloads of the activations and the data payloads of the targets of a constraint; temporal conditions constraining the time distance between activation and target [6].

Example 2 represents a constraint that can be expressed with MP-Declare.

1.3.2. RuM

RuM [3] allows the user to design MP-Declare models, discover declarative models from process execution logs, generate logs from an MP-Declare model and perform conformance checking between process execution logs and MP-Declare models. Figure 1 depicts the manual process model creating-editing menu with the selection of templates, activities and data/time conditions. In addition to the highlighted *MP-Declare Editor*, the selection bar also displays *Discovery*, *Conformance Checking*, *Log Generation* and *Monitoring*. The other options allow the user to respectively generate declarative models from log files, perform conformance checking, produce logs for a process model and perform conformance checks at run time on ongoing process executions.



Absence[A]
Absence2[A]
Absence3[A]
Alternate Precedence[A, B]
Alternate Response[A, B]
Alternate Succession[A, B]
Chain Precedence[A, B]
Chain Response[A, B]
Chain Succession[A, B]
Co-Existence[A, B]
Exactly1[A]
Exactly2[A]
Existence[A]
Existence2[A]
Existence3[A]
Init[A]
Not Chain Succession[A, B]
Not Co-Existence[A, B]

Activation: A

A does not occur

Positive examples: BC, BBCCDD

Negative examples: A, BCDAA

Figure 1: RuM [3]

1.4. Text mining

Text mining connects techniques of information retrieval, information extraction and natural language processing with the algorithms of knowledge discovery in databases, data mining, machine learning and statistics to analyse text using machines [8]. As such, the definition of text mining depends on the specific related research area; text mining as text data mining focuses on extracting patterns and information from text.

1.4.1. Text encoding

Large data sets of text documents need to be preprocessed and converted into more suitable data structures before mining can be performed. During preprocessing, the *tokenization* process converts a text document into a stream of words which can then be *filtered* by removing unnecessary words (e.g. articles, conjunctions) and finally *lemmatized* by mapping verbs to the infinitive tense and nouns to the singular form or *stemmed* by constructing the basic forms of the words [8].

The text document can then be represented by a group of words (bag-of-words representation) [8], which contains the list of present words, including their frequency of appearance. Example 3 presents the bag-of-words representation of the sentence “Alice is playing while Bob is thinking” in JSON form.

Example 3: {“Alice”:1,“is”:2,“playing”:1,“while”:1,“Bob”:1,“thinking”:1}

Another way of representing the document is to use a vector representation, in which each word is assigned a numerical *importance* value; a common implementation of this solution is the vector space model, which represents documents as vectors in *m*-dimensional space [8]. Documents are described by a numerical feature vector $w(d) = (x(d, t_1), ..., x(d, t_m))$ and as such, they can be compared by using basic vector operations; by encoding query terms in a query vector, it is even possible to perform a query by comparing the query vector to the documents and ordering the result list based on the computed similarity [8].

At this point, it is possible to apply text mining methods without any additional preprocessing. In some cases, however, further preprocessing may be used to enhance the accessible information [8]: *part-of-speech tagging* detects the part-of-speech tags for terms (e.g. verb, noun, adverb, etc.); *text chunking* groups adjacent words in a sentence (e.g. noun phrases); *word sense disambiguation* resolves the vaguenesses behind single words (e.g. homographs); *parsing* produces a full parse tree of a sentence, which can be used to find relations between words or the word’s function in a sentence (e.g. subject, verb, etc.).

2. Proposed Approach

The proposed Speech2RuM approach allows the user to use speech input in addition to the Graphical User Interface interaction. The user will be able to build declarative process models using three main functions: using speech input to define constraints with natural language; expanding constraints with multi-perspective conditions; editing and joining the constraints of the model.

This chapter outlines the conceptual contributions to the first two functions; transforming natural language speech into declarative constraints and expanding the constraints with multi-perspective conditions.

2.1. Constraint generation

The starting point for the Speech2RuM approach is a speech input, which is encoded into natural language text using a black box speech recognition library. Google Cloud Speech-to-Text was used as the speech recognition library. Then, to interpret this text, an approach created by van der Aa et al. [9] is used for the automatic extraction of constraints from natural language text. This approach supports the extraction of Init, End, Precedence, Response and Succession templates and their negated forms. In this project, this approach was extended in order to handle more flexible inputs, cover eight additional templates, and support data and time conditions. The tool can be found at https://github.com/hanvanderaa/declareextraction.

2.1.1 Linguistic processing

During the first step of the constraint extraction, linguistic processing is performed to identify the *semantic components* from the constraint description that enable the differentiation between distinct constraint templates. Figure 2 gives an overview of the components.

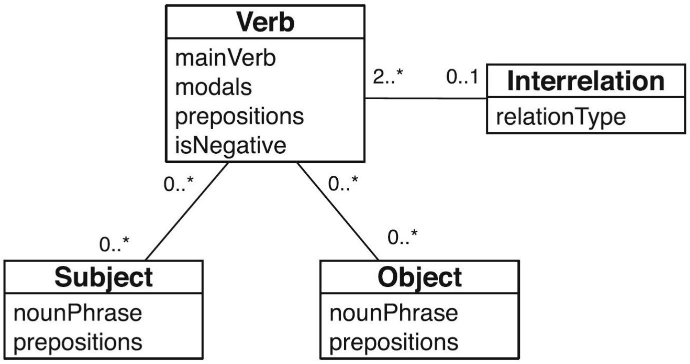


Figure 2: Semantic components extracted from linguistic processing [9]

Verbs are used to express actions in natural language texts, e.g. "*to walk*". They can be effectively identified by using part-of-speech taggers.

The *subject* of a verb specifies the entity performing the action (e.g. a manager *actor* creating a claim), while the *object* of a verb specifies the entity acted upon (e.g. a claim *business object*). The subjects and objects are identified by computing *dependency grammars* using natural language processing techniques, which reveal the grammatical relationships among words using dependency relations [9]. The approach uses Stanford relations for extracting subjects and objects [9]. The *nsubj* relation is used to extract most of the subjects and denotes the subject performing a verb. The *dobj* relation denotes the direct object of a verb, the *nsubjpass* denotes the synthetic subject in a passive clause; both are used for extracting objects.

The extracted verbs, objects, subjects are further enhanced with *specifiers*, in particular modal verbs, negations and prepositions. *Modal verbs* are extracted from related verbs and are used to identify certain constraints, e.g. Response and Succession [9]. They are auxiliary verbs indicating feasibility (mainly "*can/could*", "*may/might*", "must", "*will/would*", "*shall/should*"). *Negations* are identified through the negative dependency relation and are used to identify negative constraints. *Prepositions* express relationships between nouns and other parts of a sentence; for example, the term "*before*" can be used to mark the ordering of actions.

The approach also extracts information about two types of *interrelations* existing among verbs in the constraint description. *Adverbial clauses* are dependent clauses modifying other entities in the text; in constraint descriptions, they can indicate relations between verbs, e.g. temporal relations. *Coordinating conjunctions* express relations between entities, particularly in descriptions containing multiple constraints.

The resulting semantic components are used to identify the activities present in the constraint description.

2.1.2 Activity extraction

Activities describe *actions* performed on a *business object*, possibly by an *actor*, present in process descriptions and as such symbolize the most important constructs in a process description [9]. Activities can be either verb-based or noun-based.

Verb-based activities are constructed from the semantic components extracted during the previous processing step, where the verb stands for the action, its subject for the actor and its object to the activity's business object. As an example, the sentence "*manager processes a claim*" translates to the activity *process claim*, with *manager* as the actor.

Noun-based activities appear in constraint descriptions as noun phrases which correspond to process activities, e.g. "*processing of a claim*". In order to extract them, the approach focuses on identifying verbs which describe the process flow, *temporal verbs* such as "*to start*", "*to proceed*", "*to end*", "*to follow*", "*to happen*" [9]. The object and the subject of the verb are used to generate an activity.

2.1.3 Additional constraints

Declarative constraints are then generated using the extracted activities and their semantic interrelations. The approach was extended in order to support eight additional constraint types.

The **Participation(a)** constraint states that activity *a* must take place. It is identified by the activity list containing only one activity *without* the negation specifier. A natural language description is given in Example 4.

Example 4: An invoice must be created.

The **Absence(a)** constraint states that activity *a* must not take place. It is identified by the activity list containing only one activity *with* the negation specifier. A natural language description is given in Example 5.

Example 5: A cheque must not be refused.

The **AtMostOne(a)** constraint states that activity *a* must not take place more than once. It is identified by first recognizing the Participation constraint, after which the description is checked for a specification on the number of executions (e.g. "*at most once*", "*one time*"). A natural language description is given in Example 6.

Example 6: An invoice should be paid at most once.

The **CoExistence(a,b)** constraint states that if activity *a* takes place, then activity *b* must take place and vice versa. It is identified by the activity list containing two activities in an *and* relation, extracted from a coordinating conjunction. A notion of obligation should also be present. A natural language description is given in Example 7.

Example 7: Order shipment and invoice payment should occur together.

The **RespondedExistence(a,b)** constraint uses logic similar to the Response template, but does not specify an order for the actions. It is identified by the activity list containing two actions in a dependency relation where the target action is specified to be mandatory. A natural language description is given in Example 8.

Example 8: If a product is produced, it must be tested.

The **NotCoExistence(a,b)** constraint states that activities *a* and *b* must not both take place. It is identified as the negated version of CoExistence and RespondedExistence constraints, as both of their descriptions state that if one of the actions has taken place, the other one must not take place. A natural language description is given in Example 8.

Example 9: If an application is accepted, it cannot be rejected.

The **ChainPrecedence(a,b)** constraint states that action *b* can only occur if it is directly preceded by activity *a*. It is identified by first recognizing the Precedence constraint, after which the description is checked for a temporal preposition specifying the immediacy of the action (e.g. “*immediately*”, “*instantly*”, “*directly*”). A natural language description is given in Example 10.

Example 10: After an order is received, it may immediately be refused.

The **ChainResponse(a,b)** constraint states that if action *a* occurs, then activity *b* must immediately occur after. It is identified by first recognizing the Response constraint, after which the description is checked for a temporal preposition specifying the immediacy of the action (e.g. “*immediately*”, “*instantly*”, “*directly*”). A natural language description is given in Example 11.

Example 11: After an order is received, it must directly be checked.

2.2 Multi-Perspective conditions

The Speech2RuM approach supports adding data and time perspectives to declarative constraints, effectively turning Declare constraints into MP-Declare constraints. Three types of conditions are supported through speech input: *activation*, *correlation* and *time* conditions.

The descriptions of conditions express more textual fragments rather than full sentences, unlike descriptions of constraints; what is more, as they are short statements, their variance is much lower than that of descriptions of declarative constraints. As such, the approach uses pattern matching to extract conditions instead of grammatical parsing used during constraint generation.

Activation conditions define requirements that must be met for a constraint to be activated, e.g., a constraint is activated only if a numeric field associated with the activation is greater than a set number.

Complex conditions are supported by concatenating multiple logical statements: an input string is first split into substrings by detecting coordinating conjunctions (*and* and *or*) and splitting the string accordingly. Each substring is expected to match an expression. Expressions are then joined with logical operators. Pattern matching is supported on both numerical and categorical attributes (Table 5).

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 activation condition patterns

Correlation conditions must be adhered to when the target of a constraint is being executed, expressing relations that must exist between attributes of the activation and target action. The approach supports two patterns, either that the attribute should be the same (e.g. "*The color is the same*") or different (e.g. "*The user is different*") for the activation and the target of a constraint.

Time conditions specify the time elapsed between the activation and the target of a constraint, e.g., that the target should occur between 2 to 6 days after the activation. Three general patterns are supported (Table 6), with the first one specifying only the upper bound of the duration. Time conditions specified using seconds, minutes, hours and days are accepted.

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 time condition patterns

3. Implementation

This chapter gives an overview of the implementation of the speech recognition component and the supported functionalities of the tool.

3.1. Speech recognition implementation

Existing voice recognition libraries were compared, analysing the accuracy of the decoding, the requirement for network connection, the flexibility of setting up language models and the integration into the RuM project. After reviewing these attributes, Sphinx4 and Google Cloud Speech-To-Text libraries were inspected more closely in order to make a final decision.

Sphinx4 [10] is a pure Java open source speech recognition library, belonging to the CMUSphinx toolset. It works offline, requiring no internet connection or license or credentials. It also supports creating a dictionary or even an entire language model.

Google Cloud Speech-to-Text [11] libraries support speech recognition via applying neural network models to audio. The Java client library was explored. The encoding of the Google Cloud solution was very accurate, but needed internet access, as it worked over server requests, and required credentials which were used for validation.

The Google Cloud Speech-to-Text solution was chosen for its accuracy, which overbalanced the negative aspects of requiring credentials and an internet access. An application was built which starts a connection with the Google server using the Java application programming interface (API) when prompted and returns the first recognised transcribed sentence string. The application can be found at https://github.com/Baldre/Speech2RuM.

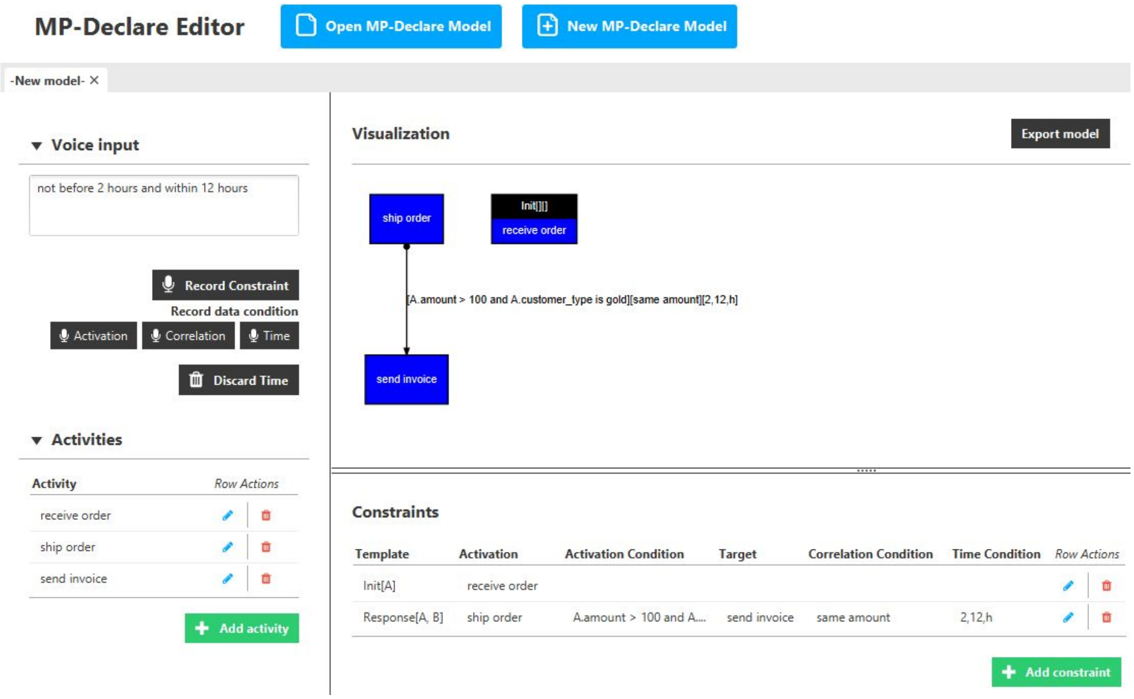
During this development the RuM dependency system was also moved over to the Maven framework in order to resolve conflicts between components and facilitate further development.

3.2. Functionalities

The speech recognition and constraint extraction components were integrated into the RuM project and were implemented in Java 11. The project can be found at https://bitbucket.org/doorless1634/thesis/commits/branch/speech-2-rum or in a packaged version at https://tinyurl.com/wjbhb4x.

The flow for creating a constraint using speech recognition with the Speech2RuM approach is as follows (from the Speech2RuM Help File, Appendix I):

1. The user presses the "Record Constraint" button.
2. The tool will now listen for the next input sentence from the microphone. It is possible to cancel the recording mid-way by pressing the "Cancel" button that appears.
3. Once the tool has recognized a sentence, it will display the sentence in the text box above the "Record Constraint" button, and will add the detected activities and constraints to the corresponding areas.
4. If needed, it is now possible to edit the recognized sentence in the text box. By editing the recognized sentence, the resulting activities and constraints are also changed. It is also possible to discard the recognized sentence entirely, removing all of the last generated constraints and activities. The user can edit the activities by clicking the button to the right of each activity, or the constraints by clicking the button to the right of each constraint. While editing, it is possible to change the wording of the actions, the constraint type, or add conditions to constraints. Data and time conditions can also be added via speech recognition using the corresponding recording button: "Activation", "Correlation", and "Time". Also these conditions can be modified or discarded.
5. The user can repeat Steps 1 to 4 to create additional constraints.



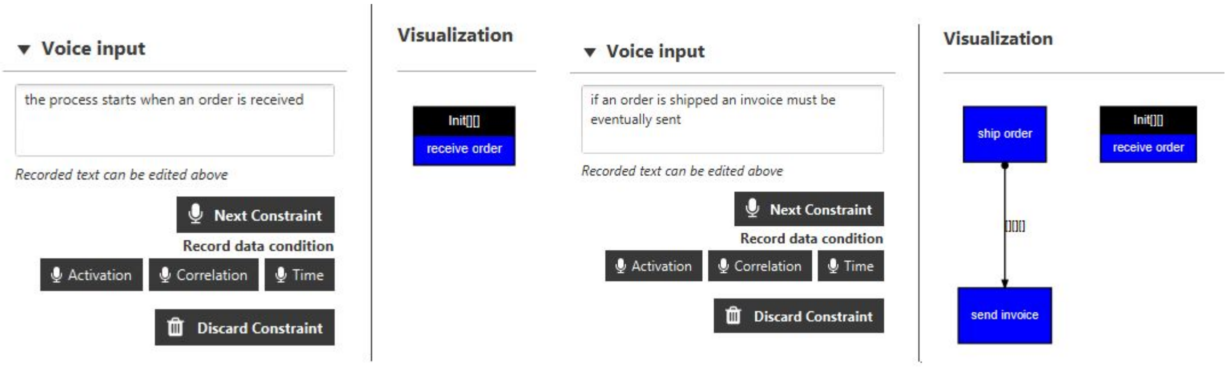
not before 2 hours and within 12 hours

Activity	Row Actions
receive order	
ship order	
send invoice	

[A.amount > 100 and A.customer_type is gold][same amount][2,12,h]

Template	Activation	Activation Condition	Target	Correlation Condition	Time Condition	Row Actions
Init[A]	receive order					
Response[A, B]	ship order	A.amount > 100 and A....	send invoice	same amount	2,12,h	

Figure 3: Interface of the Speech2RuM implementation



the process starts when an order is received

Recorded text can be edited above

if an order is shipped an invoice must be eventually sent

Recorded text can be edited above

(a) Inserting an Init constraint

(b) Inserting a Response constraint

Figure 4: Inserting control-flow properties

Figure 3 displays a screenshot of the latest version of the tool. The top-left area of the screen shows the recognized vocal input. The graphical notation of the constraints is expressed in the top-right area of the screen, with the modifiable lists of activities and constraints in the bottom part of the screen. In Figure 4, constraints are being modeled. A time condition is being recorded in Figure 3 ("*not before 2 hours and within 12 hours*"), with an activation condition ("*amount* is greater than 100 and *customer type* is *gold*") and a correlation condition ("the *amount* attribute is the same for the activation and target") already added to constraints.

The help file also gives an overview of the supported constraint types, data and time conditions, with input formatting guidelines, corresponding logic explanation and examples.

4. User Evaluation

In order to improve the existing implementation and evaluate the utility of the Speech2RuM approach, a qualitative user study was conducted with the aim of (1) finding ways to improve the interface, (2) analyze issues and needs of users from different backgrounds, and (3) discover possible usage scenarios. This chapter describes the study followed by the findings and the potential threats to its validity.

4.1. Study

For the study, eight participants were selected with different characteristics (P1 to P8 in Table 8). Two participants were selected for their experience related to BPM (tier 1); two for their background in temporal logics (tier 2); two for having used Declare for modeling temporal constraints (tier 3); and two participants for having used different tools to model, analyze and execute Declare models (tier 4). The different backgrounds were used for differentiation in order to study how users with different levels of expertise perceive the tool and recognize possible issues and needs.

The study was conducted over Skype. To provide an easily understandable context for the study, a bicycle factory manufacturing process scenario was used:

> *Whenever the sales department receives an order, a new process instance is created. A member of the sales department can then reject or accept the order for a customized bike. In the former case, the process instance is finished. In the latter case, the warehouse and the engineering department are informed. The warehouse immediately processes the part list of the order and checks the required quantity of each part. If the part is available in-house, it is reserved. If it is not available, it is back-ordered. This procedure is repeated for each item on the part list. In the meantime, the engineering department prepares everything for the assembly of the ordered bicycle. If the warehouse has successfully reserved or back-ordered every item of the part list and the preparation*

activity has finished, the engineering department assembles the bicycle. Afterwards, the sales department ships the bicycle to the customer.

A help file was also created to give an overview of the tool's functionalities and to aid the user in defining constraints (Appendix I).

Before the study took place, an email was sent to the participants with the link to the scenario, the help document, the consent form (Appendix II) and the tool, advising them to read the documents and to run the tool before the study took place. The study began with the introduction of the study procedure and the tool. The participants were then given an overview of the consent form and a verbal consent was asked. The participants sent a signed copy of the consent form after the study. Before moving onto the test procedure, a demonstration of the tool was given, during which a constraint was generated using the voice input.

The participants started the test scenario by sharing their screen which was then recorded. They were guided through the prepared test scenario, which consisted of constructing declarative process models. The shared screen was used to note down any issues or suggestions that the participants might point out during the study, as they were suggested to "think aloud" by making clarifying statements. They were first asked to insert constraints using preconstructed sentences, in order to get them accustomed to the tool. The following sentences were presented for the first assignment:

1. "The process starts when an order is received"
2. "After receiving an order, it may be accepted"
3. "Available parts must be reserved"

They were then asked to formulate constraints in natural language, given a semi-formal description of the constraint. The second assignment used the following examples:

1. Before activity "notify the warehouse", mandatory to have activity "accept order"
2. After activity "check quantity", mandatorily execute activity "order wheels"
3. Immediately after activity "notify the warehouse", mandatorily execute activity "check part list".

4. Activity “accept order” and activity “reject order” cannot occur for the same instance.

At the end of the study, a follow-up interview was conducted, focusing on resolving questions about the issues the participants had encountered. The participants were asked what they *“liked about the tool”*, *“wished was different”*, *“under which circumstances they would use it”*. The study concluded with a post-survey questionnaire (Table 7) consisting of the System Usability Scale [12] and scales covering satisfaction, expectation confirmation and usefulness, adapted from [13]. The complete questionnaire can be found at https://git.io/Jv1HC. The length of the studies ranged from 14 to 35 minutes.

Scale	Items
System Usability Scale (based on [12]), anchored between strongly disagree and strongly agree.	I think that I would like to use Speech2RuM frequently. I found Speech2RuM unnecessarily complex. I thought Speech2RuM was easy to use. I think that I would need the support of a technical person to be able to use Speech2RuM. I found the various functions in Speech2RuM were well integrated. I thought there was too much inconsistency in Speech2RuM. I would imagine that most people would learn to use Speech2RuM very quickly. I found Speech2RuM very cumbersome to use. I felt very confident using Speech2RuM. I needed to learn a lot of things before I could get going with Speech2RuM.
Perceived satisfaction (based on [13]), anchored between 1 and 5.	(1) Very dissatisfied to (5) Very satisfied (1) Very displeased to (5) Very pleased (1) Very frustrated to (5) Very contented (1) Absolutely terrible to (5) Absolutely delighted
Expectation confirmation (based on [13]), anchored between strongly disagree and strongly agree.	My experience using Speech2RuM was better than what I expected. The functionality and usability provided by Speech2RuM was better than what I expected. Overall, most of my expectations towards using Speech2RuM were confirmed.

Future use intentions (based on [13]), anchored between strongly disagree and strongly agree.	I intend to continue using Speech2RuM rather than not continue using it. My intentions are to continue using Speech2RuM rather than any other tool to model constraints. If I could, I would like to continue using Speech2RuM as much as possible.
Perceived usefulness (based on [13]), anchored between strongly disagree and strongly agree.	Using speech input improves my performance when modeling constraints. Using speech input increases my productivity when modeling constraints. Using speech input enhances my effectiveness when modeling constraints. Overall using speech input is useful when modeling constraints.

Table 7: Questionnaire scales

The collected data was analyzed by first creating an affinity diagram [14] based on the observations, follow-up interviews and video recordings by associating each reported issue with a single paper note. The notes were then clustered based on the appearing themes, resulting in 139 notes over 21 clusters. The data extracted from the questionnaire was also used as a qualitative data point during the analysis (Table 8).

	all	**tier 1 (P6, P7)**	**tier 2 (P3, P8)**	**tier 3 (P2, P4)**	**tier 4 (P1, P5)**
SUS	73.13	71.25	81.25	67.50	72.50
Satisfaction	3.46	3.67	3.67	3.00	3.50
Expectation	3.69	3.67	3.50	3.50	4.00
Future intentions	2.61	2.33	2.00	2.00	3.67
Usefulness	3.00	2.63	3.13	2.75	3.50

Table 8: Questionnaire results as means across different target groups

4.2. Findings

Compared to the commonly reported average SUS score of 68 [12], the average score of 73.13 (Table 8) reveals the tool to be reasonably usable. However, the usability issues revealed during the study also allowed us to improve the tool (Figure 3); the main changes in the improved version include the ability to undo changes applied to the model based on the last recording, editing the recorded text directly instead of needing to record the text again if wrongly recognized and adding data conditions using voice input instead of adding them manually.

As a common observation, it was found that all participants wrote sentences down before saying them out aloud. They wrote the sentences down *"on paper"* (P1), *"typed them on [their] computer"* (P2), or noted down key parts of what they wanted to say (P7). The sentences the participants used for voice input seemed somewhat unnatural; they used activity names instead of natural sentences (*"after [activity1], [activity2] is executed"*, P7). Some tried to identify keywords that the algorithm might recognize (*"is 'activity' like a keyword?"*, P8). Only P1 used natural sentences throughout the study. It might have been possible to push the participants to use more natural sentences by asking them to describe a familiar process, but this would have also affected the comparability of the findings.

Adding support for keywords might have increased the complexity of entering constraints, which would have been counterproductive, since the tool was designed with the aim of understanding natural language expressions. These issues rather indicate that users need to learn how to use speech input as a way to model constraints, regardless of having previous experience with Declare or BPM. This further implies that users should preferably go through a training phase in order to learn how to work with the tool and understand all of its functionalities.

4.2. Impact of different participant backgrounds

The different backgrounds of the different participants also affected the results of the study. Participants from tiers 1 and 2 (low confidence with Declare) relied mainly on the visual process model to evaluate the result of the voice input, whereas tier 3 and 4 participants (more familiar with Declare and Declare-based analysis tools) relied mainly on the list of semi-structured entered constraints (Figure 3, bottom-right). This was demonstrated by the way they approached possible errors; participants of tiers 1 and 2 initially tried to edit the visual process model (*“the activity label should be editable”*, from P6, P2) - not possible in the version of the tool used during the testing; participants from tiers 3 and 4 started editing the constraint list straight away (P1, P5). Only P2 attempted both.

A common issue the users had was the way the recognized text was displayed (Figure 3, text field below *Voice Input*). Participants checked the correctness of the input after they had entered the first constraint, but did not do so afterwards (P3, P6). Especially among less experienced participants, this gave the impression that they had not formulated a constraint correctly, when the problem was a wrong recognition of the voice input. A possible way to mitigate the problem would be to draw the user’s attention to the updated text field, e.g. by highlighting.

The findings indicate that even if users can bypass interacting with the graphical notation of Declare constraints by using the speech input, it is still difficult for less experienced users to evaluate the correctness of the entered constraints. This issue could be addressed by allowing the users to mark constraints that they are unable to verify, so they can then ask a more experienced user to assess their validity. In addition, traces containing examples and counterexamples of the recorded behaviors could be presented so that the correctness of the entered constraint could be easily verified.

The previous issues might have also resulted in the less experienced participants regarding the tool as less useful and being less willing to use the tool again than the experienced participants (Table 8). This finding is counterintuitive to the fact that experienced participants were slightly less satisfied with the tool than less experienced participants, but which can nevertheless be potentially explained by the experienced participants having higher initial expectations regarding the functionalities of the tool (*“what about recording data conditions as well?”*, P5).

It should be noted that participants from different backgrounds were *“excited”* (P5) to use the tool. Some continued recording constraints after the evaluation had ended (*“let me see what* *happens when I [...]"*, P8). It can therefore be reasoned that addressing the diagnosed issues could improve the perception of the tool.

4.3. Potential usage scenarios

The participants were asked for scenarios in which they would consider using speech input for entering constraints. Most considered speech input to improve their efficiency as it would allow them to *"quickly input stuff"* (P4), notably when reading *"from a textual description"* (P1); P5 also noted that it would be useful for *"people that are not familiar with the editor"*. As a contrasting opinion, P8 considered that *"manually is quicker"*, however acknowledging that speech input could be useful *"when I forget the constraint name"*. Participants also pointed out that the voice input might be useful *"for designing"* (P3), suggesting that the tool is beneficial in the early stages of the modeling process. P1 mentioned that the voice input would be useful in mobile scenarios, e.g. *"using a tablet"*, as entering text in such scenarios is more time consuming when using a mechanical keyboard.

4.4. Threats to validity

The goal of the study was to diagnose ways to improve the tool, potential usage scenarios, and issues users from diverse backgrounds might face. As such, an in-depth qualitative study was conducted with participants from a diverse range of backgrounds in context of their experience with BPM, temporal logic and Declare in particular. Using a small sample size is accepted, as the number of additional insights gained deteriorates drastically per added participant [15]. Possible threats to the study design are, however, that a specific tool was designed, which was then tested on a specific group of people in a specific environment over a limited amount of time.

Even though the group of participants was chosen carefully and the testing environment resembled an environment in which the tool is expected to be used, it is not possible to generalize the findings beyond the context of the study, as it could result in a different outcome when conducted with different participants or using a different setup.

5. Related Work

The aim of this project was to make the creation of process models as accessible as the creation of textual documents by allowing users to apply their own way of expressing Declare constraints. A case study conducted by van der Aa et al. [16] showed that using textual documents for capturing process specifications is already valued by organizations, as they focus more on the non-technical information when compared to process models, which are regarded more as technical documentation. The case study mainly focused on fragmentation of process information in organizations, determining it to be caused by the lack of standardization and guidelines, making it harder to find information in the available documentation in a way that becomes harder to keep it up-to-date.

As textual documents can be created by almost anyone and are easily accessible, Friedrich et al. [17] have created an approach for extracting BPMN models from such documents. Although it lacked an empirical user study, the authors implemented anaphora resolution (the identification of the concepts which are referenced using pronouns) and a manual preprocessing step to avoid noise. Another approach for extracting BPMN models presented in [18] demonstrated high acceptability of the method and the minimization of effort felt by the participants, though the approach did not cover the complete set of the language's notation. Schumacher et al. [19] present a workflow extraction framework with three anaphora resolution approaches to support the development of other workflow extraction applications by a pipes-and-filters architecture.

In addition to extracting formal models from textual documents, van der Aa et al. [20] have proposed an approach for using textual process specifications for model verification, by generating an alignment between the activities of the process model and the sentences of the textual description, after which missing activities and conflicting ordering can be detected by using predictors. Sànchez-Ferreres et al. [21] present a solution that goes even further, providing optimal alignments between the nodes of a process model and the sentences of the textual description, aligning multiple process model elements, e.g. events and gateways in addition to activities.

Another work by van der Aa et al. [22] applies textual documents for direct process analysis, implementing the concept of *behavioral space*, which captures all the possible interpretations of the process description in order to deal with ambiguities, as well as using a semi-automated pruning technique to reduce the remaining level of uncertainty. Sànchez-Ferreres et al. [23] also propose a new multi-perspective language, ATDP (Annotated Textual Description of a Process), used for representing processes based on textual annotation. The language captures the ambiguity of natural language by unifying all the valid interpretations of the process.

As for extracting declarative process models from natural language sentences, the preliminary work on the extraction of Declare constraints from texts [9] forms the foundation of this work by providing a state-of-the-art approach which we have extended. Another rule-based approach was presented by Selway et al. in [24], which extracts SBVR models from natural language and was motivated by the contrast between the needs of business people, who prefer natural language descriptions for business models, and those of technical experts, who need formal models for information systems development, which can be used for automated process analysis (e.g. conformance checking).

6. Conclusion

In this thesis, we have presented an interactive approach for taking vocal input from the user and applying speech recognition and natural language processing to extract multi-perspective declarative process models from it. The Speech2RuM approach surpasses the state-of-the-art in text-to-constraint transformation by supporting a larger variety of Declare templates, as well as permitting their augmentation with data and time conditions. By integrating the approach into the RuM toolkit, the user is able to visualize and edit the obtained models in a GUI. What is more, this allows the user to use the models directly within the toolkit to apply its analysis techniques, e.g. conformance checking and log generation. Finally, it must be noted that the conducted user evaluation represents the first study into the feasibility of using speech recognition in business process modeling. The results demonstrated the approach's promising nature, as well as a clear learning curve.

The Speech2RuM approach can be further developed based on the gathered user feedback. The text-to-constraint component can be improved by supporting less natural descriptions, for example keywords such as *activity*, which would denote a process step. It would also be beneficial to support the specification of multi-perspective constraints in a single step rather than two. In the end, it would be fascinating to explore how the speech recognition component could be used in creation of other types of process models, such as procedural process models.

7. References

[1] Weske, M. (2007). Business Process Management–Concepts, Languages, Architectures, Verlag. *Berlin*.

[2] Pesic, M. (2008). Constraint-based workflow management systems:shifting control to users. Technische Universiteit Eindhoven. https://doi.org/10.6100/IR638413

[3] Kapisiz, D. Rule Mining with RuM, *Masters Thesis, University of Tartu*, 2019.

[4] Van Der Aalst, W. (2011). *Process mining: discovery, conformance and enhancement of business processes* (Vol. 2). Heidelberg: Springer.

[5] Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., ... & Weber, B. (2016). Understanding declare models: strategies, pitfalls, empirical results. *Software & Systems Modeling*, *15*(2), 325-352.

[6] Burattin, A., Maggi, F. M., & Sperduti, A. (2016). Conformance checking based on multi-perspective declarative process models. *Expert Systems with Applications*, *65*, 194-211.

[7] Maggi, F. M., Mooij, A. J., & van der Aalst, W. M. (2011, April). User-guided discovery of declarative process models. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)* (pp. 192-199). IEEE.

[8] Hotho, A., Nürnberger, A., & Paaß, G. (2005, May). A brief survey of text mining. In *Ldv Forum* (Vol. 20, No. 1, pp. 19-62).

[9] Van der Aa, H., Di Ciccio, C., Leopold, H., & Reijers, H. A. (2019, June). Extracting declarative process models from natural language. In *International Conference on Advanced Information Systems Engineering* (pp. 365-382). Springer, Cham.

[10] Shmyrev, N. (n.d.). Building an application with sphinx4. Retrieved May 7, 2020, from https://cmusphinx.github.io/wiki/tutorialsphinx4/

[11] Cloud Speech-to-Text - Speech Recognition | Google Cloud. (n.d.). Retrieved May 7, 2020, from https://cloud.google.com/speech-to-text/

[12] Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, *189* (194), 4-7.

[13] Bhattacherjee, A. (2001). Understanding information systems continuance: an expectation-confirmation model. *MIS quarterly*, 351-370.

[14] Holtzblatt, K., Wendell, J. B., & Wood, S. (2005). Rapid contextual design: a how-to guide to key techniques for user-centered design. *Ubiquity*, *2005* (March), 3-3.

[15] Nielsen, J., & Landauer, T. K. (1993, May). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems* (pp. 206-213).

[16] Van der Aa, H., Leopold, H., van de Weerd, I., & Reijers, H. A. (2017). Causes and consequences of fragmented process information: Insights from a case study.

[17] Friedrich, F., Mendling, J., & Puhlmann, F. (2011, June). Process model generation from natural language text. In *International Conference on Advanced Information Systems Engineering* (pp. 482-496). Springer, Berlin, Heidelberg.

[18] De Oliveira, J. P. M., Avila, D. T., dos Santos, R. I., & Fantinato, M. (2017, November). Assisting Process Modeling by Identifying Business Process Elements in Natural Language Texts. In *Advances in Conceptual Modeling: ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6–9, 2017, Proceedings* (Vol. 10651, p. 154). Springer.

[19] Schumacher, P., Minor, M., & Schulte-Zurhausen, E. (2013, August). Extracting and enriching workflows from text. In *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)* (pp. 285-292). IEEE.

[20] Van der Aa, H., Leopold, H., & Reijers, H. A. (2017). Comparing textual descriptions to process models–the automatic detection of inconsistencies. *Information Systems*, *64*, 447-460.

[21] Sànchez-Ferreres, J., van der Aa, H., Carmona, J., & Padró, L. (2018). Aligning textual and model-based process descriptions. *Data & Knowledge Engineering*, *118*, 25-40.

[22] Van der Aa, H., Leopold, H., & Reijers, H. A. (2018). Checking process compliance against natural language specifications using behavioral spaces. *Information Systems*, *78*, 83-95.

[23] Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., & Padró, L. (2019, September). Formal Reasoning on Natural Language Descriptions of Processes. In *International Conference on Business Process Management* (pp. 86-101). Springer, Cham.

[24] Selway, M., Grossmann, G., Mayer, W., & Stumptner, M. (2015). Formalising natural language specifications using a cognitive linguistic/configuration based approach. *Information Systems*, *54*, 191-208.

Appendix

I. Speech2RuM Help File

Extracting Declare Constraints from Natural Language

This document describes the functionality of the Speech2RuM modeling tool. Section 1 describes the steps that users should take to specify, extend, and adapt declarative process constraints. Section 2 specifies basic constraint types supported by the tool. Section 3 describes enriched versions of these types, e.g., using negation or chain constraints, whereas Section 4 describes how constraints can be augmented with data and time conditions.

1. VOICE TOOL FLOW

To enter voice control menu:

1. Enter the “MP-Declare Editor” view and select “New MP-Declare Model”.

Creating a constraint using the speech tool:

1. Press “Record”.
2. The tool will now listen for the next input sentence from the microphone. It is possible to cancel the recording mid-way by pressing the “Cancel” button.
3. Once the tool has recognised a sentence, it will display the sentence in the text box above the “Record” button, and will add the detected activities and constraint(s) to the corresponding areas.
4. If needed, it is possible to edit the recognised sentence in the text box or the activities by clicking the button to the right of each activity, the same for constraints. While editing, it is possible to change the wording of the actions, the constraint type, or add conditions to constraints (see Section 4). By editing the recognised sentence, the result activities and constraints are also changed.
5. The user can repeat Steps 1 to 4 to add additional constraints.

2. SUPPORTED BASE CONSTRAINT TYPES

The current implementation supports nine base constraint types: INIT, EXISTENCE, ABSENCE, PRECEDENCE, RESPONSE, SUCCESSION, COEXISTENCE, RESPONDED EXISTENCE, AT-MOST-ONCE. This section describes these types and the input required to express them in natural language.

INIT - Defines the activity that should happen first for a process instance.

Input req.: A sentence that contains a process-related noun, e.g., “process”, “instance”, “case”, and a verb denoting that something starts, e.g., “start”, “begin”.

Examples:

- Init(customer arrives) - “The process starts when the customer arrives”
- Init(receive order) - “A case begins when an order is received”

EXISTENCE - Defines an activity that should occur at least once per process instance.

Input req.: A sentence describing a single activity that includes an obligation, e.g., “must”, “should”, “will”.

Examples:

- Existence(create an invoice) - “An invoice must be created”
- Existence(each product pass a test) - “Each product should pass a test”

ABSENCE - Defines an activity that should never occur.

Input req.: A sentence describing a single activity including a negation.

Examples:

- Absence(fail tests) - “Tests should not be failed”

- Absence(allow dogs) - “Dogs are not allowed in the restaurant”

PRECEDENCE - A Precedence(target, activation) constraint defines that an activity (activation) can only occur if another activity (target) happened beforehand.

Input req: A sentence that invokes an order between two activities, for which the target activity is not mandatory, e.g., not indicated with “can”, “may”, “could”.

Examples:

- Precedence(create a claim, approve a claim) - “A claim must be created before it is approved”
- Precedence(create a claim, approve a claim) - “If a claim is approved, then it must have been created first”

RESPONSE - A Response(activation, target) constraint defines that if some activity happens (activation), another activity (target) must be executed later.

Input req.: A sentence that invokes an order between two activities, for which the target activity is mandatory, e.g., indicated with “must”, “should”, “will”, and which indicates order, e.g. with “then”, "after", "later".

Examples:

- Response(ship order, send invoice) - “When an order is shipped then an invoice must be sent.”
- Response(close building, staff leaves) - “The staff must leave after the building is closed.”

SUCCESSION - A Succession(activation, target) constraint defines that an activity (activation) can only happen if and only if another activity (target) happens later.

As SUCCESSION constraints are syntactic sugar to express the conjunction of a precedence and a response, we have proposed a way to create such constraints by creating a base RESPONSE/PRECEDENCE constraint and changing the constraint type.

CO-EXISTENCE - A Co-Existence(activation, target) constraint defines that two activities should always be executed together for a process instance.

Input req.: A sentence that describes two actions that should co-occur.

Examples:

- Co-Existence(order shipping, invoice payment) - “Order shipping and invoice payment should occur together”

RESPONDED EXISTENCE - A RespondedExistence(activation, target) constraint defines that if an activity (activation) occurs, then another activity (target) must occur.

Input req.: A sentence that specifies two activities, for which the target activity is mandatory, e.g., indicated with “must”, “should”, “will”, and which does not indicate order

Examples:

- Responded Existence(produce product, test the product) - “If a product is produced, the product must be tested.”
- Responded Existence(identify an incident, log the details) - “When an incident is identified, the details must be logged”

AT-MOST-ONCE / ABSENCE2 - defines that a certain activity should be executed at most once per process instance. Displayed in RuM as “Absence2”.

Input req.: A sentence describing a single activity that indicates an execution limit.

Examples:

- AtMostOnce(invoice paid) - “An invoice should be paid at most once”

- AtMostOnce(invoice paid) - “An invoice should not be paid more than once”

3. FURTHER CONSTRAINT TYPES

NOT-SUCCESSION - defines two activities that cannot succeed each other.

Req: A sentence describing two activities where the activation action must be negative (include “not”)

Examples:

- Not Succession(an error appear, we react) - “if an error appears, we will not react”
- Not Succession(clear the results, notify the employee) - “after the results are cleared, the employee is not notified”

NOT COEXISTENCE - defines two activities that cannot cooccur in the same process instance.

Examples:

- Not Coexistence(accept application, reject application) - “if an application is accepted, it cannot be rejected”

CHAIN-Constraints - use the logic of the original constraint, but the activation and target action must take place immediately after each other (no other action can happen in between)

Req: RESPONSE/PRECEDENCE constraint, where the target action is linked to an “immediate” modal. NOT and CHAIN constraints can be used together.

Immediate modals: “immediately”, “instantly”, “directly”, “promptly”

Examples:

- Chain Precedence(receive the notification, display the results) - “after the notification is received, the results are displayed immediately”
- Not Chain Response(an error appear, we react) - “if an error appears, we will not react instantly”

ALTERNATE-Constraints - use the logic of the original constraint, but the activation/target action cannot reoccur before the constraint is satisfied

As ALTERNATE constraints don't usually appear in natural language (e.g. "if action A happens, then it does not happen again before action B happens"), we have proposed a way to create such constraints by creating a base RESPONSE/PRECEDENCE constraint and changing the constraint type.

4. CONDITION TYPES

Conditions can be added to a constraint after the constraint type and actions have been set.

ACTIVATION CONDITION - Specifies a constraint on the activation activity.

Req: numerical condition: field is (not) (greater, lower than (or equal to)) value; categorical condition: field (not) in list of values. Multiple conditions are separated by "and" or "or".

Examples:

- "A.value > 4" - "value is greater than four"
- "A.type not in (basic, medium, large)" - "type is not in basic, medium, large"

CORRELATION CONDITION - specifies a constraint that considers a field of both the activation and target activity.

Req: same/different field. Also accepted are conditions in the Activation Condition format. Multiple conditions are separated by "and" or "or".

Examples:

- "same price" - "price is same"
- "different resource and A.orders == 5" - "source is different and orders is five"

TIME CONDITION - specifies a constraint that considers the time elapsed between the activation and target action

Req: between X and Y seconds/minutes/hours/days

Examples:

- “5,13,m” - “between 5 and 13 minutes”
- “0,2,d” - “at most 2 days”
- “10,15,s” - “not before 10 and no later than 15 seconds”

II. Speech2RuM Consent Form

Consent to Act as a Participant in a Research Study

Study title: Speech2RuM evaluation

Principal Investigator: Karl Johannes Balder

Co-Investigators: Alexander Nolte, Fabrizio Maria Maggi, Han van der Aa

Introduction: As an experienced process analyst you were selected to participate in this study. This is a joint study conducted by a research group from the University of Tartu (Estonia), the Free University of Bozen-Bolzano (Italy) and Humboldt University of Berlin (Germany).

The aim of the study is to evaluate Speech for RuM - a tool that supports speech input for modeling constraints. We will particularly focus on identifying problematic aspects of the current version of Speech for RuM and means for improvement.

During the study we will ask you to use Speech for RuM to model constraints in the context of two scenarios that you have received via email. For each of these scenarios we will ask you to first model 3 predefined constraints and then add 3 constraints of your choice.

We will record your interaction with Speech for RuM and our conversation during the study. After the study we will ask you a few follow-up questions in a short interview, and we will ask you to fill out a short survey. The study overall will take no more than 60 minutes of your time.

Participation requirements: Any person 18 or older who has experience modeling constraints and that is a fluent English speaker is eligible to participate.

The expected duration of the study: The study will take about 45-60 minutes of your time.

Risks and Benefits: The risks that are associated with this research are no greater than those ordinarily encountered in daily life. There are no direct benefits to participants, but the researchers anticipate future improvements to the Speech for RuM tool to potentially benefit process mining researchers and practitioners.

Privacy and Confidentiality: In order to protect the participants' identities during this study the research team will follow the following procedure: The original recordings will only be accessible to the Principal and Co-Investigators. The video files will be used to analyze the interaction of the participant with the Speech for RuM tool. Audio contained in the recordings will be transcribed, potential identifiers will be removed or aggregated and the original recordings will be deleted afterwards.

Your data and consent form will be kept separate. Your consent form will be stored securely and will not be disclosed to third parties.

By participating, you understand and agree that the data and information gathered during this study may be used by the participating universities for publication purposes. However, any identifiable information will not be mentioned in any such publication or dissemination of the research data and/or results. The University of Tartu requires all research records to be maintained for at least 5 years following final reporting or publication of a project. Aggregated data will thus be archived by the Principal Investigator for that timespan.

Questions about the Study: If you have any questions, comments, or concerns about the study either before, during, or after participation, please contact the Principal Investigator (Karl.Johannes.Balder@tudeng.ut.ee) or the Co-Investigators.

Voluntary Participation: Your participation in this research is voluntary. You may discontinue participation at any time during the research activity. Your decision regarding whether or not to participate in this study will not result in any loss of benefits to which you are otherwise entitled.

I am of age 18 or older. I have read and understood the information above and I want to participate in this study:

☐ Yes ☐ No

Participant: The above information has been explained to me and all of my current questions have been answered. I understand that I am encouraged to ask questions, voice concerns or complaints about any aspect of this study during its course, and that such future questions, concerns or complaints will be answered by a qualified individual or by the investigator(s) listed on the first page of this consent document.

Co-investigator: I certify that I have explained the nature and purpose of this research study to the participant, and I have discussed the potential benefits and possible risks of study participation. Any questions the participant had about this study have been answered, and we will always be available to address future questions, concerns or complaints as they arise. I further certify that no research component of this study has started before this consent form was signed.

Participant____________________________ Co-investigator____________________________

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I,

Karl Johannes Balder,

(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Defining Business Process Models with Natural Language Processing and Speech Recognition,

   (title of thesis)

   supervised by Fabrizio Maria Maggi.

   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Johannes Balder

08/05/2020