

Freie Universität Berlin

Bachelor's thesis at the Software Engineering Research Group of the Institute of
Computer Science at Freie Universität Berlin

Development of a BPMN Model Validation and Automatic Correction Tool for SAP Solution Manager

Rainier Raymond Robles

Student ID: 5002400

ren@inf.fu-berlin.de

First Examiner: Prof. Dr. Lutz Prechelt

Second Examiner: Prof. Dr. Claudia Müller-Birn

Berlin, September 4, 2019

Abstract

The goal of this thesis is the creation of a tool that assesses Business Process Model and Notation (BPMN) models created with SAP Solution Manager (SolMan) according to readability and usability guidelines. Necessary corrections are performed based on this assessment, resulting in a new BPMN file that can be imported back into SolMan. SAP Solution Manager is an Application Life-cycle Management (ALM) product used in the organization and maintenance of a company's business and IT operations. BPMN is a standard for the graphical representation of business processes in a process model. While SolMan provides a tool for creating process models with BPMN, it does not provide a function to validate these models, unlike similar tools in the market. After research and selection of guidelines, creation of a proof-of-principle prototype, and construction of the product using an iterative and incremental development model, a functional web application that is able to review and make corrections according to seven guidelines was successfully developed.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Affirmation of independent work

I hereby declare that I wrote this thesis myself without sources other than those indicated herein. All parts taken from published and unpublished scripts are indicated as such.

Berlin, September 4, 2019

Rainier Raymond Robles

Contents

Acknowledgement	7
1 Introduction	9
1.1 Motivation	9
1.2 Goal	9
1.3 SAP Solution Manager	10
1.4 Business Process Model and Notation	10
1.5 Vostura GmbH	12
1.6 Outline	13
2 Related Work	15
3 Solution Approach	17
3.1 Selection of Guidelines to be Implemented	17
3.1.1 Selection Criteria	17
3.1.2 Guidelines Selected	19
3.2 Architecture	24
3.2.1 Implementation Environment	24
3.2.2 Integration of Existing Libraries or Ground-Up Development . .	25
3.3 Implementation Strategy	26
3.4 Proof-of-Principle Prototype	26
3.4.1 Goals	26
3.4.2 Challenges	27
3.4.3 Results	27
4 Implementation and Evaluation	29
4.1 Iteration 1: Initial Implementation	29
4.1.1 Goals	29
4.1.2 Challenges	29
4.1.3 Results	29
4.1.4 Evaluation	30
4.2 Iteration 2: Modularization	30
4.2.1 Goals	30
4.2.2 Challenges	31
4.2.3 Results	32
4.2.4 Evaluation	33
4.3 Iteration 3: Design Polishing and First Usability Tests	33
4.3.1 Goals	33
4.3.2 Challenges	34
4.3.3 Results	35
4.3.4 Usability Tests and Evaluation	35

4.4	Iteration 4: Model Visualization and Second Usability Tests	37
4.4.1	Goals	37
4.4.2	Challenges	37
4.4.3	Results	38
4.4.4	Usability Tests and Evaluation	38
5	Conclusions	43
5.1	Summary of Results	43
5.2	Suggested Future Work	44
5.3	Personal Lessons	46
	Bibliography	47
A	Appendix	49
A.1	Questions of the Usability Tests	49
A.2	Models Used for the Usability Tests	50

Acknowledgement

First of all, I would like to thank my thesis supervisor, Prof. Dr. Lutz Prechelt, for the assistance and advice provided during the period of this thesis, from its conception until its completion, for both the big picture as well as the little details.

I would also like to show gratitude to Vostura GmbH for entrusting me with this topic and providing on-site support for this project. In particular, I would like to thank Eric Siegeris and Philipp Claus for the feedback and guidance throughout this process.

I could not finish this work without a little help from my friends around the world. Special thanks go to Sergelen Gongor and Christian Zygmunt Jeschke in Berlin and Ginoo Karlo Galvez Tan in the Philippines for reading early drafts of my thesis and giving helpful notes on how to improve it; my Freie Universität Berlin classmates and friends Sergelen, Christian, Ingrid Tchilibou, and Hong Zhu for not just being willing participants in my tests, but also providing moral support; Luis Merino García and Hendrik Niemann in Berlin for being good sounding boards and providing vital input; Fingal Olsson and the rest of the Olsson family in Sweden for their kindness, allowing me to relax, recharge, and stay motivated halfway through the process; and Albert Černý, Jeroným Šubrt, and Antonín Hrabal in Czechia for keeping me going thanks to good music and making me feel like I'm not just a friend of a friend.

Lastly, none of this would be possible without the love and support from my family in the Philippines, particularly my mother.

1 Introduction

1.1 Motivation

Companies that use **SAP Solution Manager** (SolMan) to organize and document their businesses have the option to use a tool to model their business processes graphically using **Business Process Model and Notation** (BPMN). This has the dual purpose of providing a visual representation of their processes as well as a means of documenting their individual steps and, when applicable, linking them to the technical systems and procedures (such as SAP transactions) utilized for that particular activity.

As those tasked with business process modeling within an organization may not necessarily be experts in BPMN, it would be advantageous to provide a means to check the syntactic correctness as well as the readability and usability of the models created. The modeling tool in SolMan does not include model validation, a feature found in most commercial BPMN modeling software. Offering an integrated model validation tool would help reduce the amount of time spent in the maintenance and correction of user-created models.

The topic of this thesis was proposed to me by **Vostura GmbH**, an SAP consulting company whose clients includes companies that use SolMan to create BPMN models. I saw this as a good opportunity to be able to provide a worthwhile service for novice BPMN modelers. As someone who was once a novice BPMN modeler, I knew from first-hand experience that it was difficult at times to concentrate on the finer details of the model when working on larger processes.

1.2 Goal

The primary goal of this thesis is to create a working prototype for an integrated BPMN model validation tool for SolMan. In this case, "integrated" means that the user does not need to download additional software or use a non-SAP website to check a BPMN model created using SolMan. Instead, the tool will be a web-based program that can be opened directly in SAP.

In addition, the capability to automatically correct violations of the validation criteria will be built into the tool, providing clients with clear, understandable models. This will save them time otherwise spent manually updating their models. It is also hoped that usage of the tool will increase users' BPMN understanding and proficiency, reducing future errors.

This application will be implemented with diagrams designed in and exported from SolMan in mind. These diagrams have particular characteristics which need to be handled accordingly, such as how gateway and sequence flow labels are exported compared to other diagram modeling programs.

The intended users of this tool are employees of companies that use Solution Manager to model their business processes with BPMN. Based on personal experience, they are likely to be BPMN novices prone to making errors when modeling.

1. Introduction

1.3 SAP Solution Manager

SAP Solution Manager (SolMan)¹ is an Application Lifecycle Management (ALM)² product used in the organization, maintenance, and optimization of a company's business and IT operations. One feature included in SolMan is Process Management, which allows companies to describe and document processes in a way that is useful for both IT specialists as well as business practitioners within the organization. One way processes are documented in SolMan is through creating process models in BPMN.

1.4 Business Process Model and Notation

Business Process Model and Notation (BPMN) [1] is a standard for the graphical representation of business processes in a business process model. BPMN is maintained by the Object Management Group (OMG), which also maintains other standards such as Unified Modeling Language (UML)³ and Object Constraint Language (OCL)⁴.

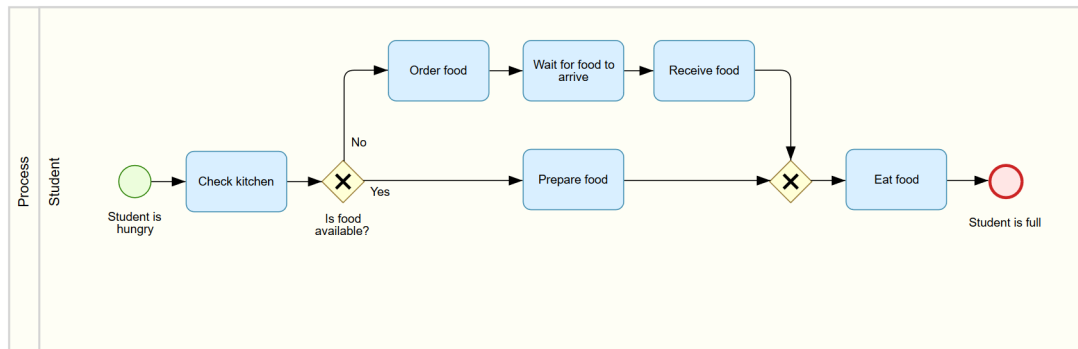


Figure 1: Example of a BPMN diagram

The BPMN standard organizes processes into **pools** which represent participants, such as a company or a specific department in the company. A pool is then separated into **lanes** which represent specific roles among the participants. For example, if a pool is assigned to the Human Resources Department, then the lanes may include Human Resources Manager, Human Resources Assistant, and Recruiter. Other graphic elements in BPMN include **events**, **gateways**, **flows**, and **activities**. Table 1 shows a selection of some commonly-used elements and what they represent in a diagram. Diagrams can be modeled horizontally or vertically.








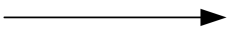

¹<https://support.sap.com/en/alm/solution-manager.html> [Accessed: 2019-04-22]

²ALM is the process of managing the lifecycle of computer programs from inception to implementation and maintenance.

³UML is a general-purpose modeling language for the creation of diagrams to help visualize a system's design.

⁴OCL is a language which describes rules that apply to UML models.

Table 1: Common BPMN graphic elements, with images from [1]

Element Name	Description	Graphic Representation
Pool	A pool represents a participant.	
Lane	A lane is a partition in a pool that organizes and categorizes activities. They can be used to represent specific roles, systems, and departments in the participant.	
Start Event	A start event indicates where a process begins.	
Intermediate Event	An intermediate event represents something that "happens" within a process that affects its flow.	
End Event	An end event indicates where a process finishes.	
Exclusive Gateway	A gateway controls how paths within a process branch or merge. An exclusive gateway marks alternative paths, where only one path can be taken. Also known as an XOR Gateway.	
Parallel Gateway	A parallel gateway creates and combines parallel paths. Also known as an AND Gateway.	
Sequence Flow	A sequence flow represents the order in which activities are performed in a process.	
Activity	An activity represents work that is performed in a process.	

In addition to its graphic representation, a BPMN model also has a corresponding XML⁵ file. It is organized with a tree structure, with parent and child elements. Figure 2 shows more details of the XML tree associated with BPMN files exported from SolMan.

⁵XML (Extensible Markup Language) is a markup language which defines rules for documents that have to be both human- and machine-readable.

1. Introduction

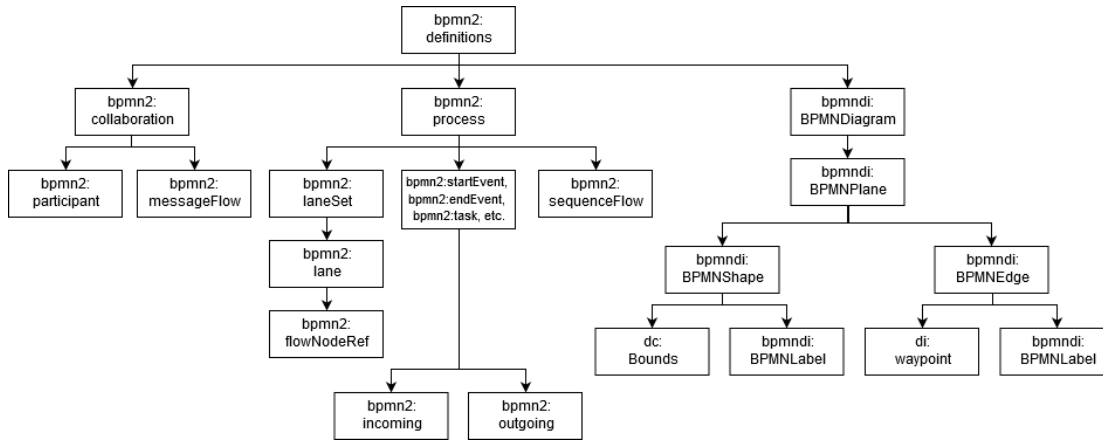


Figure 2: Partial XML tree for BPMN files from SolMan

The **bpmn2:definitions** element is the root element of the BPMN file. This element has at least three child nodes:

- One **bpmn2:collaboration** node, which has a **bpmn2:participant** child node for each pool in the model. This section organizes the different participants that are part of the process portrayed.
- **bpmn2:process** nodes for each pool in the diagram. Each has a **bpmn2:laneSet** child element, which in turn has a **bpmn2:lane** child for each lane in that particular pool. References to the BPMN elements contained in that lane are declared as **bpmn2:flowNodeRef** children. The actual BPMN elements are declared individually as children of the **bpmn2:process** node they are contained in, with each BPMN element having its own tag. For example, activities are **bpmn2:task** elements, while sequence flows are **bpmn2:sequenceFlow** elements.
- One **bpmndi:BPMNDiagram** node, which includes details for the visual rendering of the BPMN diagram. It has a **bpmndi:BPMNPlane** child node, which in turn has a **bpmndi:BPMNEdge** child node for every sequence flow and a **bpmndi:BPMNShape** child node for every other BPMN graphic element.

Every element of a BPMN diagram corresponds to multiple nodes in its associated BPMN file. For example, an activity corresponds to a **bpmn2:flowNodeRef** child of the **bpmn2:lane** node it is contained in, a **bpmn2:task** node as a child of the **bpmn2:process** node it is contained in, and a **bpmndi:BPMNShape** node for rendering purposes.

1.5 Vostura GmbH

Vostura GmbH⁶ is a small IT consulting company based in Berlin and Mannheim specializing in SAP consulting, particularly with ALM and SAP Solution Manager. Vostura supports clients who elect to model their business processes within SolMan using SAP's integrated modeling tool. The company offers a training course in order

⁶<http://vostura.com/> [Accessed: 2019-04-22]

to introduce the basics and conventions of BPMN modeling to clients. Additional support is also offered in the form of modeling processes with SolMan that can serve as examples, or reviewing BPMN models created by their clients.

During the period of this thesis, I was employed by Vostura as a working student.

1.6 Outline

In Section 2, I review available related literature that proposes readability and usability guidelines for BPMN models, and how existing tools support these guidelines. In Section 3, I discuss how I plan to implement the tool. I talk about major decisions such as the guidelines to be implemented and the architecture of the software. In Section 4, I go in-depth into each implementation iteration, discussing the goals, challenges, and results, as well as evaluating the product at the end of each iteration. Finally, in Section 5, I summarize the progress done on creating the application and discuss suggestions on how to expand on the work completed through this thesis. I also reflect upon the process and share my personal learnings from this experience.

2 Related Work

There are different methods and criteria to assess the quality of a model. A framework proposed by Lindland, et al. [2] identifies three types of model quality. **Syntactic quality** refers to how well the model corresponds to the specification or language it is created in. The goal is to completely adhere to the specification. **Semantic quality** refers to how closely the model corresponds to reality. The model is supposed to accurately and completely represent the problem or process it is supposed to portray. Lastly, **pragmatic quality** refers to how well users can interpret the model. The goal here is to make the model easier to understand by the model's audience. As assessing the semantic quality of a BPMN model requires prior knowledge of the business process that it is supposed to represent, I decided to focus on guidelines that tackle either the syntactic or the pragmatic quality of a model.

Guidelines regarding syntactic quality based on the BPMN 2.0 specification currently exist. Studies have determined over 600 constraints based on the specification [3][4], which another study formalized as 755 OCL expressions [5]. Examples of constraints found in these studies are: A start event should not have any incoming sequence flows; and, if an implicit start event is used, meaning the start event symbol is not used, then an implicit end event should be used, meaning the end event symbol is not used, and vice versa. Apart from the sheer number of guidelines involved, one problem with using this as a basis for my thesis is that many constraints refer to attributes which BPMN files of models exported from SolMan do not have. For example, elements can be described by the number of outgoing or incoming sequence flows that it has. If an element has more than one sequence flow that points away from it, meaning it has multiple outgoing sequence flows, then it is said to be splitting, while if it has more than one sequence flow pointing towards it, meaning it has multiple incoming sequence flows, then it is joining. One gateway attribute defined in the specification is *gatewayDirection*, which identifies whether a gateway is splitting (when the attribute value is "diverging"), joining ("converging"), or both ("mixed" or "unspecified") [1]. This attribute is not present in SolMan BPMN files. As a result, there may be a high number of violations to these syntactic constraints due to the way SolMan exports BPMN files. Guidelines that do not apply to SolMan BPMN files would therefore need to be filtered out. Moreover, studies have pointed out inconsistencies and lack of clarity within the specification [6]; for example, an attribute may be marked as optional in a UML class diagram but considered mandatory in the corresponding table [3]. Such inconsistencies would also need to be considered.

A paper by De Boeck and Claes [6] pinpoints a smaller subset of 26 syntax issues that cover the most frequently used BPMN elements as identified by Zur Muehlen and Recker [7]. These guidelines include the absence of explicit start or end events, and missing sequence flows between activities. This provides a good starting point for including guidelines regarding syntactic quality to the tool.

Improving the usability and understandability of BPMN models is one of the goals of this thesis. Thus, it is important not only to focus on guidelines regarding syntactic

2. Related Work

quality, but also those regarding pragmatic quality. The list of 26 syntax issues mentioned earlier includes what it calls *confusions*, where something may be syntactically correct based on the specification but is considered "bad practice" as it degrades the pragmatic quality of the model. Examples of confusions include multiple sequence flows coming from or going towards non-gateway elements, and the use of multiple start or end events [6].

A technical report by Moreno-Montes de Oca and Snoeck [8] based on a systematic literature review [9] resulted in a total of 56 individual guidelines meant to improve the pragmatic quality of BPMN models. These guidelines were classified under the following three categories:

1. Guidelines that count elements. This category includes guidelines such as: A process model should not have more than 31 elements; and, there should not be more than 3 sequence flows coming from a gateway.
2. Morphology. This focuses on the form of the model in terms of loops, parallelism, etc. This category includes guidelines such as: Avoid loops in the process model; and, avoid a high degree of parallelism.
3. Presentation. This focuses on stylistic aspects of the model that do not affect its semantics. Guidelines regarding labels are included, as well as guidelines regarding overlapping lines and the alignment of various elements.

Snoeck, et al. [10] then studied how many of the guidelines they put forward are tested and supported by various BPMN tools in the market. They found that the highest level of support in any product was for 32 of the 56 guidelines, while the lowest level of support was for 11 guidelines. The study takes into consideration various levels of support, from Forced Support, wherein the user is forced to follow the guideline and fix the problem before proceeding, to Documentation, wherein the guidelines are found in the tool's documentation but not in the tool itself. When guidelines that are supported only through documentation are not considered, the highest level of support dips to 27 out of 56, while the lowest level of support goes down to 9 guidelines.

Corradini, et al. [11][12], meanwhile, formulated a list of 50 guidelines to ensure understandable BPMN models. While many overlap with the aforementioned 56 guidelines, some go into more detail. For example, while Moreno-Montes de Oca and Snoeck simply state that gateways should be labeled, Corradini and associates specify which gateway types should and should not be labeled. Some guidelines are also unique to Corradini, et al. One example is that the default path after a splitting exclusive gateway should be expressed.

While these works focus mainly on improving pragmatic quality, some guidelines are rooted in the BPMN specification, thus establishing that some level of syntactic quality is also ensured. These guidelines therefore form the basis for my thesis.

3 Solution Approach

3.1 Selection of Guidelines to be Implemented

The 56 guidelines presented by Moreno-Montes de Oca and Snoeck [8] as well as an additional five by Corradini and associates [11][12] were evaluated in order to determine which of these would be implemented in the tool. First, I attempted to create models in SolMan that violated each guideline. If a particular violation could not be modeled, then it was thrown out, as there was no reason to check for rules that could not be broken in the first place. For example, rules which refer to the use or absence of color highlights were discarded in this step because the color scheme of diagrams created in SolMan could not be changed. Another discarded rule was regarding pools not having labels, as this was also impossible to model. Seven guidelines were discarded through this method.

Guidelines which were more subjective and could not be quantified were also taken out of consideration. In some cases, these can be covered indirectly by other existing rules. For example, the guideline against models that are too large can be covered by the guideline about the suggested maximum number of elements. Rules which would have required text analysis, such as one which discouraged the usage of abbreviations, were also disregarded, as validation checks for these rules would require machine learning, which is outside the scope of this thesis. Furthermore, guidelines which checked the same or similar properties were combined. From the initial 61 guidelines, only 40 remained after this process.

Then I manually analyzed 25 diagrams which were modeled by myself and a coworker at Vostura. These were created towards the beginning of a client project we were involved in, when we were both relatively new to BPMN. This would provide insight into which rules were most commonly broken by novice modelers. Furthermore, the guidelines were mapped to Vostura's internal BPMN model quality assurance criteria.

3.1.1 Selection Criteria

In order to select which guidelines would be implemented in the tool, the following criteria were considered:

- **Part of Internal Guidelines:** This criterion corresponds to whether or not a particular guideline is part of the Vostura guidelines for quality assurance of BPMN models. There are two possible values for this criterion: **Yes** if it's included in the Vostura guidelines, and **No** otherwise.
- **Automated Adjustment Possible:** This criterion corresponds to the possibility of providing a fix in case the guideline is violated. There are three possible values for this criterion: **Yes** means a fix can be easily provided regardless of the details of the model; **Yes with Reservation** means a fix can be provided but either the fix is incomplete (for example, in case of a missing label, only

3. Solution Approach

a placeholder label can be provided) or details of the model need to be taken into consideration (for example, changing the position of one element can have cascading effects on other elements); and, **No** means an automatic fix cannot be provided.

- **Ease of Implementation:** This criterion corresponds to the perceived ease of implementing the suggested adjustment. There are four possible values for this criterion: **Trivial** means the implementation is perceived as relatively simple without potential side effects; **Challenging** means that additional steps or qualities need to be taken into consideration in the implementation (for example, several elements may need to be moved in order to fit a new element); **Complex** means that a more complex algorithm may need to be implemented (for example, determining whether two elements overlap, then determining how to correct this); and, **Impossible** means an automatic fix cannot be implemented. Guidelines marked *No* under Automated Adjustment Possible are automatically given the label *Impossible* here.
- **QA Misses:** This criterion corresponds to how many of the 25 Vostura-created BPMN models in the client project referred to earlier violated the guidelines. There are three possible values for this criterion: **Many** means at least ten models violated the guideline; **Few** means at least one and at most nine models violated the guideline; and, **None** means no models violated the guideline.

While two of the criteria (Part of Internal Guidelines and QA Misses) are objective, quantitative measurements, the other two are generally more subjective. On one hand, differentiating between *Yes/Yes with Reservation* and *No* under the criterion Automated Adjustment Possible was objective; for example, the guidelines regarding the number of elements in the model was marked as *No*, because fixing a violation of this guideline requires combining or deleting a selection of objects, which is not possible without understanding the actual process being represented. On the other hand, there is a fine difference between marking a guideline with *Yes* or *Yes with Reservation* which is more subjective. For example, I assigned the value *Yes* to guidelines which referred to missing explicit start and end events, because adding an event before the first element or after the last element seemed relatively straightforward. On the other hand, I assigned the value *Yes with Reservation* to a guideline regarding gateways which showed both splitting and joining behavior. While the idea for the suggested adjustment, which was to create a second gateway and insert it before the existing gateway, also seemed straightforward, the fact that this would be done in the middle of a diagram and other elements had to be moved around meant that this would not be as simple as in the previous case.

Meanwhile, the criterion Ease of Implementation is based on a subjective "feel" for the complexity of the algorithm to be implemented and my own capabilities as a programmer. For example, issues with guidelines regarding missing labels could be resolved by making a single adjustment to a single element, which is why I assigned those guidelines with the value *Trivial*. Meanwhile, fixing gateways with both splitting and joining behavior required adjustments to more elements because of the need to move them. While this presented a challenge, this was an algorithm that was

comprehensible to me, hence that guideline was assigned with the value *Challenging*. Lastly, as mentioned above, I assigned a guideline about overlapping elements with the value *Complex* because there were too many cases to take into account. I had to find a way to determine which of two overlapping elements had to be moved, how to move it, and the effects that moving it had on other elements in the diagram.

All guidelines were assigned values for each of the four selection criteria. I then chose guidelines to implement based on these values. First, I selected guidelines which were labeled *Yes* for Part of Internal Guidelines, *Yes* for Automated Adjustment Possible, and *Trivial* for Ease of Implementation. These would be the first to be implemented. Then I picked those which were labeled *Yes* for Part of Internal Guidelines, either *Yes* or *Yes with Reservation* for Automated Adjustment Possible, and either *Trivial* or *Challenging* for Ease of Implementation.

Finally, guidelines that were labeled *Yes* or *Yes with Reservation* for Automated Adjustment Possible and *Complex* for Ease of Implementation, and either *Yes* for Part of Internal Guidelines or *Many* or *Few* for QA Misses were chosen. Due to the perceived complexity of the algorithms related to guidelines in this group, it was expected that implementing these would be time intensive. Thus, I discussed these with Vostura in order to determine which would be more valuable for the tool. These were then ranked according to perceived added value as determined by this discussion.

3.1.2 Guidelines Selected

The following seven guidelines were selected using the above method and implemented in the tool. A *Description* of the guideline and, in some cases, why it is important will first be given, followed by a description of the *Validation* steps involved, and finally an explanation of the *Correction* procedure.

1. Absence of Start Events

Description: Although the usage of a start event is not a requirement according to the BPMN specification [1], the lack of an explicit start event is a pragmatic error that negatively affects the understandability of a model [8][13]. The use of implicit start events could lead to misinterpretation and confusion about where and when the process starts [11]. While Moreno-Montes de Oca and Snoeck suggest an upper limit of two start events [8], Corradini et al. state in their guideline "Use start events consistently" [12] that only one explicit start event should be included.

Validation: If a start event node exists in the BPMN file, then the diagram contains an explicit start event.

Correction: The goal is to find an appropriate position for the start event to be inserted. Presumably, the model contains (at least) one implicit start event, which is an activity, gateway, or event that has no incoming sequence flows. In the event that only one element has no incoming sequence flows, then this will be identified as the implicit start event. In case there are no such elements, which suggests that the process has been modeled with a loop, then the element positioned on the leftmost side of the diagram will be named the implicit start event. In the event that there are multiple potential implicit starts, the one

3. Solution Approach

positioned on the leftmost side of the diagram will be chosen as the implicit start event. Once the implicit start event has been found, the pool and lanes of the process will be increased in size to accommodate the new elements, an explicit start event will be placed to the left of the identified implicit start event, and a sequence flow from the start event to this first element will be inserted.

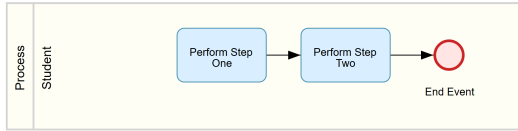


Figure 3: Absence of start events in a model

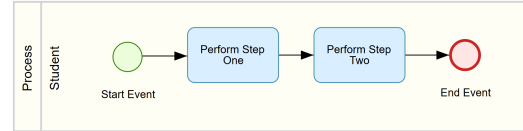


Figure 4: Same model with an explicit start event

2. Absence of End Events

Description: Analogous to start events, the explicit use of an end event is not a requirement according to the BPMN specification [1]. However, the lack of an explicit end event is a pragmatic error that has a negative effect on the understandability of a model [8][13]. It may be unclear when a process ends. Models with only one end event are found to be less error prone [8], however the use of different end events for distinct end states such as success and failure are also allowable [12]. However, it is beyond the scope of this thesis to determine whether or not numerous potential end points correspond to distinct end states, hence the tool will only insert one end event.

Validation: If an end event node exists in the BPMN file, then the diagram contains an explicit end event.

Correction: The correction is analogous to that of the previous guideline. It is assumed that the model contains (at least) one implicit end event, which is an activity, gateway, or event that has no outgoing sequence flows. In the event that one only element has no outgoing sequence flows, then this will be identified as the implicit end event. In case there are no such elements present, then the element positioned at the rightmost side of the diagram will be named the implicit end event. In case there are multiple potential implicit end events found, then the one positioned at the rightmost side of the diagram will be selected. Once the implicit end event has been found, the pool and lanes of the process will be increased in size, an explicit end event will be placed to the right of the identified implicit end event, and a sequence flow from that last element to the end event will be inserted.

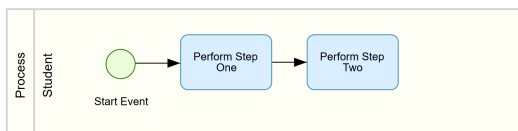


Figure 5: Absence of end events in a model

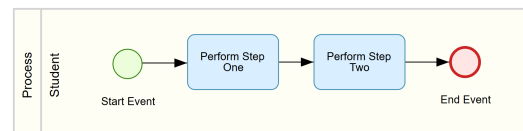


Figure 6: Same model with an explicit end event

3. Unlabeled Sequence Flows

Description: Outgoing sequence flows from splitting exclusive gateways should have labels [6][10][11][12]. As exclusive gateways are considered as questions to be asked in the process, each answer to the question is represented as an expression associated with one of the gateway's outgoing sequence flows [1]. The expression is then used as that sequence flow's label.

Validation: All outgoing sequence flows from splitting XOR gateways will be checked. If the associated sequence flow node does not have a *name* attribute, or the value of its *name* attribute is an empty string, then that sequence flow violates this guideline.

Correction: For each sequence flow node from a splitting exclusive gateway that violates this guideline, a generic string will be assigned as the value of the node's *name* attribute. If the *name* attribute does not exist, it will be created before its value is assigned.

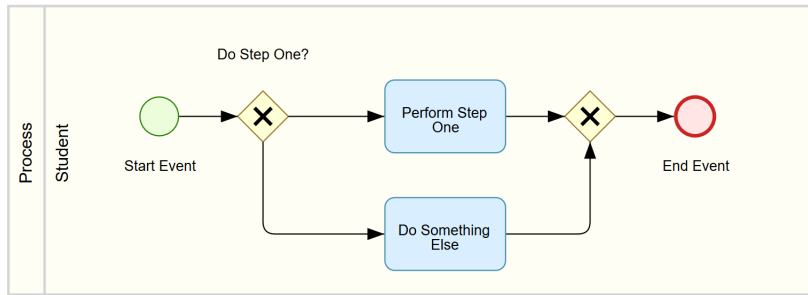


Figure 7: Unlabeled sequence flows in a model

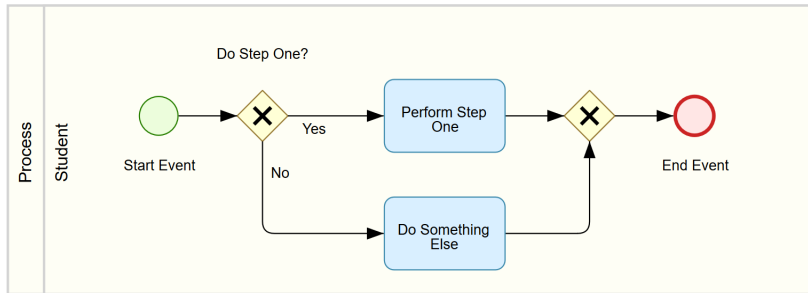


Figure 8: Same model with labeled sequence flows

4. Mixed-use Gateways

Description: As the behavior of a gateway with both multiple incoming and multiple outgoing sequence flows may be unclear [14], it is recommended that a gateway should not exhibit joining and splitting behaviors at the same time [6][8][12].

Validation: In a BPMN file, outgoing and incoming sequence flows from and to an element are represented by bpmn2:outgoing and bpmn2:incoming child nodes. If a gateway node's children includes more than one outgoing child node

3. Solution Approach

and more than one incoming child node at the same time, then that gateway violates this guideline.

Correction: The idea is to split mixed-use gateways into two distinct gateways, one that joins and one that splits. First, the pool and lanes of the diagram will be increased in size to fit the new elements. A gateway node that violates the guideline will be copied, including references to its incoming and outgoing sequence flows. All elements of the diagram that are to the left of the copied gateway node will be moved further left to fit a new gateway and sequence flow. The copy of the gateway will then be inserted to the left of its original. All references to outgoing sequence flows will be removed from the copy, while all references to incoming sequence flows will be removed from the original gateway. Then, a new sequence flow from the copy to the original will be created.

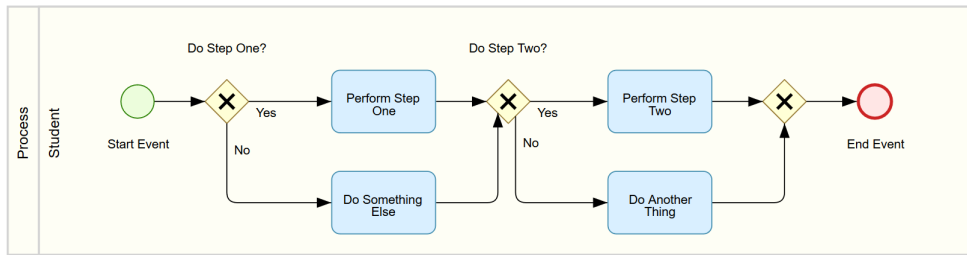


Figure 9: Mixed-use exclusive gateway present in a model

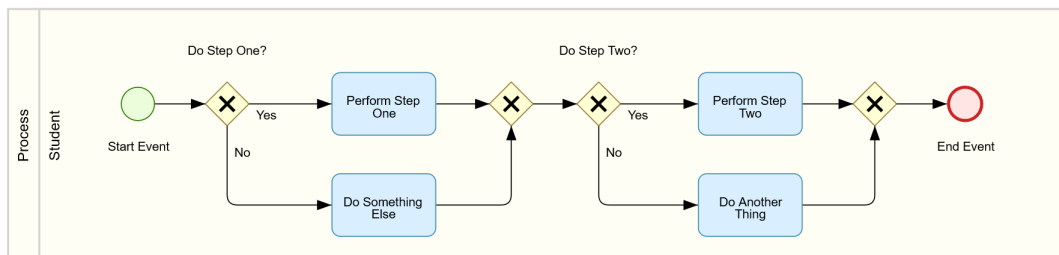


Figure 10: Same model with the mixed-use gateway now split

5. Unlabeled Gateways

Description: Moreno-Montes de Oca and Snoeck [8] mention that gateways should be labeled. Corradini, et al. [11][12] specify further which gateway types should and should not be labeled. First, splitting exclusive gateways should be labeled with a question or interrogative phrase. Next, joining gateways, particularly joining exclusive gateways, should not be labeled. Finally, both splitting and joining parallel gateways should not be labeled.

Validation: Labels of gateways in BPMN files modeled in and exported from Sol-Man are stored as values of the *name* attribute of the gateway's corresponding node. First, splitting XOR gateways, which are exclusive gateway nodes with multiple outgoing child nodes, will be checked. If the *name* attribute does not exist, or the value of the *name* attribute is an empty string, then this gateway violates the guideline. Second, joining XOR gateways, which are exclusive gateway nodes with multiple incoming child nodes, will be checked. If a joining

exclusive gateway has a label, then this gateway violates the guideline. Finally, parallel gateway nodes will be checked. If a parallel gateway has a label, then this gateway violates the guideline.

Correction: For each splitting exclusive gateway that violates the guideline, a generic string will be assigned as the value of the node's *name* attribute. If the *name* attribute does not exist, it will be created before its value is assigned. For each parallel gateway and joining exclusive gateway that violates the guideline, the value of the *name* attribute will be replaced by an empty string.

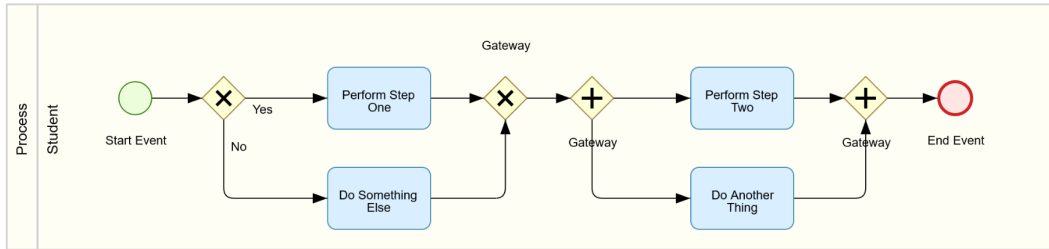


Figure 11: Model with unlabeled splitting XOR gateway, labeled joining XOR gateway, and labeled AND gateways

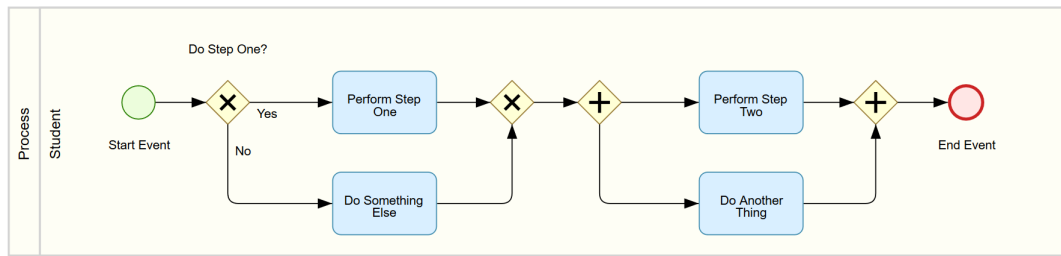


Figure 12: Same model with labeled splitting XOR gateway, unlabeled joining XOR gateway, and unlabeled AND gateways

6. Unlabeled Events

Description: Events should be labeled [10]. Corradini, et al. [11][12] specify that non-specific start and end events should not be labeled if there is only one of each in the diagram. However, Vostura's own internal quality assurance guidelines assert that any and all start and end events should be labeled, so this is what this thesis shall consider.

Validation: If an event node does not have a *name* attribute, or the value of the *name* attribute is an empty string, then the event violates the guideline.

Correction: For each event node that violates the guideline, a generic string will be assigned as the value of the node's *name* attribute. If the name attribute does not exist, it will be created before its value is assigned.

3. Solution Approach

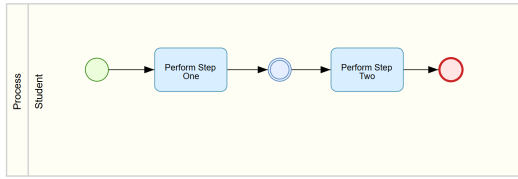


Figure 13: Unlabeled events (start, intermediate, end) in a model

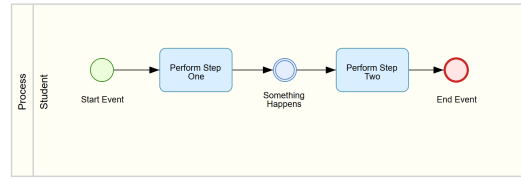


Figure 14: Same model with labeled events

7. Absence of Explicit Gateways

Description: A model that contains multiple paths is expected to include gateways [6]. To be more specific, activities or events should not be used to split or join sequence flows, and only gateways should be used to do this [11][12].

Validation: Nodes that are not of a gateway type will be inspected. If a non-gateway node contains multiple outgoing child nodes or multiple incoming child nodes, then that node violates the guideline.

Correction: The idea is to insert a splitting gateway after a non-gateway element that exhibits splitting behavior, or a joining gateway before a non-gateway element that exhibits joining behavior. First, the pool and lanes of the diagram will be increased in size to fit the new elements. If a splitting gateway is inserted, then all graphic elements to the left of the position it will be inserted at will be moved further to the left to accommodate the new gateway and sequence flow. Outgoing sequence flows from the original non-gateway element will now be coming from the new gateway, and a sequence flow from the original non-gateway element to the new splitting gateway will be created. If a joining gateway is inserted, then all graphic elements to the right of the position it will be inserted at will be moved further to the right. Incoming sequence flows to the original non-gateway element will now be pointing to the new gateway, and a sequence flow from the new joining gateway to the original non-gateway element will be created.

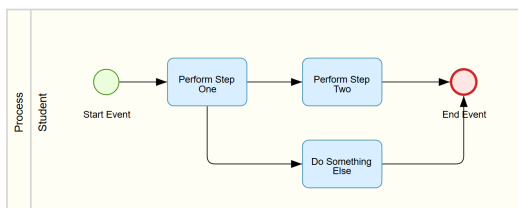


Figure 15: Absence of explicit gateways in a model

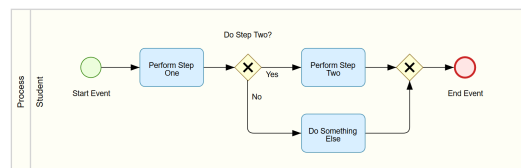


Figure 16: Same model with explicit gateways

3.2 Architecture

3.2.1 Implementation Environment

The goal is to build a program that will ultimately be integrated within the SAP ecosystem. SAP offers a means to create web-based applications that can be inte-

grated with SAP through SAP Fiori⁷ and SAPUI5⁸. While SAP Fiori dictates the design guidelines, SAPUI5 is a framework built upon HTML5, CSS3, JavaScript, and jQuery that is used to build web applications for SAP Fiori [15]. Thus, instead of implementing the application in SAP's proprietary programming language ABAP, I used JavaScript for the implementation of the guideline validation and violation correction, and HTML for the user interface, with the SAPUI5 framework serving as its base.

3.2.2 Integration of Existing Libraries or Ground-Up Development

Apart from commercial software available in the market, there are open source libraries that implement BPMN model validation. I examined some of these to determine if any of them were suitable for integration with the tool to be implemented. If so, this would save time in implementation and I could focus more on the corrections rather than on the checks.

Geiger, Neugebauer, and Vorndran [4] developed BPMNspector⁹ in order to automatically evaluate BPMN models for the over 600 constraints that they determined from the specification. However, as mentioned in *Related Work*, many of the guidelines may not be applicable due to how SolMan exports BPMN files, and culling the guidelines may end up being as time-consuming as implementing them from scratch.

Another option was the guideline validation extension for bpmn-js¹⁰. This tool inspects fewer guidelines, but these correspond to many of the guidelines I had selected for my own validation tool. However, on one hand, the extension required the use of bpmn-js [16], a BPMN rendering toolkit and modeling tool that I was not planning to use at the time, and on the other hand, perusing the source code showed that it would be unlikely to correctly validate BPMN files from SolMan. For example, different node names were used, such as bpmn:StartEvent and bpmn:EndEvent instead of bpmn2:startEvent and bpmn2:endEvent, respectively.

The final open source library I considered was BEBoP, which was developed by Corradini, et al. [12] to check their own guidelines. I created some BPMN models using SolMan which violated the guidelines to be implemented in the validation tool and tested them in BEBoP¹¹ to see if they would be correctly checked.

In the end, although BEBoP checks for most of the guidelines that would be implemented, these were also the guidelines that had been estimated to require less complex coding. In addition, some of the trickier guidelines were checked to a degree, but there were enough limitations to make building from scratch a more viable option. For example, BEBoP was unable to detect labels on gateways and sequence flows in diagrams modeled with SolMan, because SolMan exports labels differently from what BEBoP expects. As a result, validation for unlabeled gateways or unlabeled sequence flows always returned a failure despite being correctly modeled.

⁷<https://experience.sap.com/fiori/> [Accessed: 2019-08-22]

⁸<https://sapui5.hana.ondemand.com/> [Accessed: 2019-08-22]

⁹<http://bpmnspector.org/> [Accessed: 2019-08-26]

¹⁰<https://github.com/bpmn-io/bpmn-js-guideline-validation> [Accessed: 2019-08-26]

¹¹<http://pros.unicam.it:8080/BEBoP-WebUserInterfaces/contentform.jsf> [Accessed: 2019-08-16]

3. Solution Approach

Due to the limitations in the libraries I reviewed, I decided to implement the validation from scratch. This has the added advantage of allowing more flexibility for the continued development of the tool, as well as in the case that changes are made to the SolMan BPMN modeling program.

3.3 Implementation Strategy

Implementing the tool was split into two phases. In the first phase, a proof-of-principle prototype was created. Details regarding this phase follow in Section 3.4.

In the second phase, the selected guidelines were implemented using an iterative and incremental development approach. This meant that development was done through repeated cycles (iterations) in small portions of time (increments) [17]. This would allow for evaluation and improvement of what had already been programmed, as well as a way to ensure that there would be a functional prototype at the end of each iteration. Within each iteration, the goal was to use the Plan-Do-Study-Act (PDSA) Cycle in order to maximize progress within the iteration as well as to minimize defects [18]. The stages of the PDSA cycle [19] are:

1. Plan - The planning stage is where decisions are made about what to do within the cycle. In this thesis, this is where goals were set for the current iteration.
2. Do - Plans made in the previous stage are carried out in this stage. This corresponds to the actual implementation and coding of the application.
3. Study - In this stage, the results of the previous stage are assessed and compared with the plans made in the first stage. This is also sometimes known as the Check stage. In this thesis, during this stage, the functionality of the tool was tested by using the application to evaluate different models I created which violated the newly-implemented or updated guidelines in different ways. This is also where meetings with Vostura as well as usability tests were conducted.
4. Act - The process is evaluated, changes are enacted, and the next cycle begins.

Development took place over four iterations of two weeks each. At the end of each iteration, the results were presented to my bosses at Vostura, who took on the role of product owner and assessed the product and guided in which direction the next iteration of development should go. Specific details about each iteration are in Section 4.

3.4 Proof-of-Principle Prototype

3.4.1 Goals

The main goal of creating a proof-of-principle prototype was to prove that implementation of the product with the chosen architecture was feasible. If major problems arose during this stage, then the architecture would be re-evaluated and adjusted.

For the proof-of-principle prototype, the fundamental functions of the tool were implemented. First of all, one must be able to upload a BPMN model in the form of a BPMN file. Second, the tool must be able to open and read the uploaded model.

Third, the tool must be able to manipulate the BPMN file by adding, removing, or modifying nodes and attributes. Fourth, the tool must be able to produce some output based on the model. Finally, the user must be able to download the modified BPMN file.

I also used this process as an opportunity to delve deeper into the BPMN files generated by SAP Solution Manager. In order to be able to make corrections to the model, I had to understand the structure of the file, including which elements were present and what attributes each element had.

3.4.2 Challenges

The first step was determining which existing libraries, frameworks, and functions I could use in the implementation. SAPUI5 includes a FileUploader framework¹² which allows users to easily import files into SAPUI5 applications. DOMParser¹³ is also supported, which meant reading and writing to a BPMN file, as well as navigating through one, would not be a problem.

The next challenge was understanding the structure of a BPMN file. Most of the work was done by exporting and downloading a model from SolMan, reading through the source code, making minor changes, then importing the modified model back into SolMan. The most important question was whether or not elements must be declared in sequential order (from first to last element in the process flow) or spatial order (from left to right in the diagram). It was determined that the order was not relevant in correctly displaying a model in SolMan. This meant, for example, that a start event node could be added after an end event node without problems, as long as it was inserted as a child of the corresponding process node.

A major challenge was determining how to move pieces of code into functions so they could be reused, as well as for ease of maintenance. Unfortunately, none of the attempts made during the development of the prototype worked, as the tool would not run at all when external functions were used. Thus this idea was abandoned until later in the development process in **Iteration 2: Modularization**.

I also attempted to integrate a BPMN diagram rendering and visualization library [16]. My attempts were also unfortunately unsuccessful at the time, with diagrams not being rendered in the app, and this idea was abandoned until later in **Iteration 4: Model Visualization and Second Usability Tests**.

3.4.3 Results

A simple prototype with the essential functions listed under **Goals** was successfully implemented. With the help of built-in functionality within SAPUI5, users could upload BPMN files, which would then be opened, read, and manipulated using supported Web APIs such as FileReader und DOMParser. A means to download the modified BPMN file was also implemented.

The prototype worked by opening and manipulating the file as soon as it was uploaded. A means to visualize the model was needed in order to show that the file

¹²<https://sapui5.hana.ondemand.com/#/api/sap.ui.unified.FileUploader> [Accessed: 2019-08-01]

¹³<https://developer.mozilla.org/en-US/docs/Web/API/DOMParser> [Accessed: 2019-08-01]

3. Solution Approach

was successfully processed by the tool. This would be especially vital when testing the app throughout its development. Since initial attempts to render the diagram within the app using a third-party library were unsuccessful, a section of the app was allotted to display the source code of the BPMN file. If there was a problem reading or manipulating the model, then this section would be empty.

Another section of the app was allocated to display plain text based on the contents of the file. This would eventually be where the validation and correction report would be displayed, but in the proof-of-principle prototype, this section only showed a simple header as well as the name of the first lane in the uploaded process diagram. This proved that the output displayed in this section was dependent on the uploaded BPMN model.

The successful implementation of the fundamental functions of the planned tool marked the end of the planning phase and the start of the development phase.

4 Implementation and Evaluation

4.1 Iteration 1: Initial Implementation

4.1.1 Goals

For the first development iteration, the main goal was to implement as many of the chosen guidelines as possible. During this iteration, only horizontal models with one pool and one lane would be taken into consideration, because the focus was on the breadth rather than the depth of cases covered.

Another goal was to format the report. As one of the desired effects of using the tool was improved BPMN proficiency, it would not be enough to name which guidelines were checked and whether or not they were violated. The report should also describe how these violations were fixed within the model. Additional information should be provided as needed; for example, if the correction of a guideline involved adding labels to the model, then not only should the user be informed about the added labels, they should also be informed that these are generic labels that should be manually updated in SolMan.

4.1.2 Challenges

One major challenge for this iteration was the complexity of the implementation of the algorithm to fix violations of the Absence of Start Events guideline. This was the first guideline to be implemented, and although the algorithm was inherently trivial, my low familiarity of the nature of BPMN files at the time made it more challenging. The BPMN file had to be modified at multiple locations, and I would occasionally miss some, either causing the app to crash, or the updated file to be rendered incorrectly when imported into SolMan.

Another challenge involved the report. In the proof-of-principle prototype, the report took the form of a simple string. While this was adequate for displaying plain text, it did not provide any options for formatting the report. SAPUI5 provides a construct for displaying text along with HTML code¹⁴, which means that selected HTML tags could be included in the string and the app would be able to interpret them correctly. This allowed me to create headers for each guideline, as well as style and format the contents of the report. The second part of this challenge involved the text to be displayed, particularly finding the balance between providing enough information and not overwhelming the user with too much text. I decided that, for each guideline, a single sentence regarding whether or not it had been violated and a single sentence regarding any corrections made were sufficient.

4.1.3 Results

At the end of the first iteration, five guidelines were implemented: Absence of Start Events, Absence of End Events, Unlabeled Sequence Flows, Unlabeled Events, and

¹⁴<https://sapui5.hana.ondemand.com/#/api/sap.m.FormattedText> [Accessed: 2019-08-24]

4. Implementation and Evaluation

Mixed-use Gateways. The validation and correction of these guidelines only functioned on horizontal models with one pool and one lane, with the aim of adding support for multiple lanes at a later iteration.

In addition, a report with basic HTML formatting was also created. The report contained the name of the guideline, information on whether or not the guideline was violated, and, in case modification of the model was performed, details regarding the correction. For the guidelines Unlabeled Sequence Flows, Unlabeled Events, and Mixed-use Gateways, the number of violations per guideline was also included.

4.1.4 Evaluation

The results of the first iteration were presented to Vostura, including the caveat that the scope was limited. Feedback was generally positive, particularly regarding the number of guidelines already implemented. The primary criticism dealt with the layout of the app. The tool still displayed the BPMN file source code, as it did in the proof-of-principle prototype. While it served its purpose of showing the user that the BPMN file had been successfully uploaded, validated, and corrected, it was deemed unnecessary to the project, even for testing purposes, because if there were problems with uploading or processing the BPMN file, then either the app would crash, or the report would not be displayed at all. This made fixing the layout a goal for the next iteration.

Also discussed during the meeting with the company was the inability to use functions in order to reduce the amount of code, something which was discovered during the implementation of the proof-of-principle prototype. There were lines of code that were copied and pasted multiple times throughout the program across multiple guidelines, and being able to move these lines of code into separate functions within the program would improve both the readability of the code as well as its reusability, particularly in view of implementing further guidelines which required the same functions. It was agreed that this was important and should be taken into consideration in the next iteration.

4.2 Iteration 2: Modularization

4.2.1 Goals

Two goals of the second iteration were based on feedback on the results of the first iteration. These were refactoring in order to be able to move code that was used repeatedly into separate functions, and finalizing the layout of the tool without displaying the BPMN file source code.

Another goal was continued implementation of further guidelines, namely Unlabeled Gateways and Absence of Existing Gateways.

Finally, all implemented guidelines would be extended such that models with a single pool but multiple lanes could also be validated, thus increasing the depth of the cases covered.

4.2.2 Challenges

The primary challenge in this iteration was in relation to the creation of separate functions in order to reduce the amount of repeated code. These functions could then be used in the implementation of further guidelines. Attempts to create and use separate functions had thus far failed repeatedly. It seemed that because the validation and correction of the model were being done as soon as it was uploaded, it was not possible to properly call these external functions. However, when upload and validation were performed independently of each other, then calling external functions worked. Thus it was decided to separate uploading a model from validation and correction. Once this had been done, it became possible to create reusable functions.

I also encountered challenges while implementing the guideline Unlabeled Gateways. It was possible to provide labels to parallel gateways in SolMan. In fact, this is the default behavior of the SolMan modeling tool when a new gateway is added. When exporting said model into a PDF or SVG¹⁵ file, the labels would be displayed. However, when testing the exported BPMN file in my application, it always reported that the parallel gateways were unlabeled. Inspecting the BPMN file showed that labels on parallel gateways were not being exported by SolMan. In fact, when importing the file without any changes into SolMan, these labels disappear. Thus, it was unnecessary to check for labels on parallel gateways, so I removed the implementation of this particular sub-guideline. I initially considered including a message regarding this rule in the report, but I decided against it as it may end up being too confusing for users.

In addition, labels past a certain length are truncated in SolMan. In this case, instead of creating a label directly on the gateway, a text annotation can be used to label it. This method was used for some BPMN diagrams from the Vostura client project I was involved in. Thus, when checking a splitting XOR gateway for a label, the presence of a text annotation attached to the gateway also had to be checked. As events could also be labeled in this manner, this check was added to the Unlabeled Events guideline. I then discussed with my bosses at Vostura whether text annotations on joining exclusive gateways as well as on parallel gateways should be considered as labels and therefore deleted. We decided that they should be left alone because there may be other reasons for adding those particular text annotations apart from labeling.

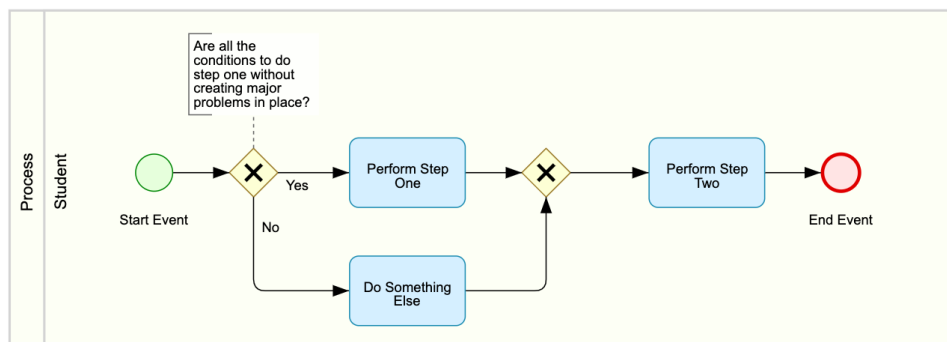


Figure 17: BPMN model with a text annotation used as a gateway label

¹⁵Scalable Vector Graphics (SVG) is a vector image format for 2D graphics that is based in XML.

4. Implementation and Evaluation

The next challenges surfaced while implementing the Absence of Explicit Gateways guideline. The first was determining what kind of gateway should be added. The choice was between adding an exclusive gateway and adding a parallel gateway. On one hand, the exclusive gateway was the default choice when adding a gateway in SolMan, and the ability to add labels to both the gateway and, in the case of splitting gateways, the sequence flows made it a good option. On the other hand, experience showed that parallel gateways were less common than exclusive gateways, so perhaps they would be more prominent when added to a model. In addition, parallel gateways and their associated sequence flows did not require labels, which would simplify the implementation and reduce clutter in the corrected diagram. I decided to use parallel gateways because of these advantages.

The other challenge associated with implementing the guideline involved manipulating the graphic elements of the diagram in order to fit the inserted gateways. This was already a challenge I had encountered in the previous iteration. However, those corrections only dealt with adding a single element and a single sequence flow. Also, when fixing violations of the Mixed-use Gateways guideline, the pre-existing gateway was copied, meaning that sizes and relative positions remained the same. For this particular case, because the element that displayed splitting or joining behavior could be one of many possible types, such as an activity or an event, the newly-created gateway would be inserted into the position of elements of varying sizes. The main issue was moving the ends of the sequence flows pointing to or from the new gateway such that they would be within the new element. When testing my implementations, there would be cases when the arrowheads of the sequence flow did not end up within the area that the new gateway covered. This led to the sequence flow not being connected to the newly-added gateway when importing the corrected model into SolMan. I decided that the best way to fix this was to make sure that the endpoint of such sequence flows was the middle point of the gateway. While this was successful in most cases, there were still a handful of instances where this was not done correctly. In the end, I could not determine the cause of these errors, nor could I determine a common pattern for these cases.

4.2.3 Results

The tool no longer automatically performed the validation and correction actions upon uploading a BPMN model. Instead, a button must be pressed after uploading. This was done in order to take advantage of the principles of modularization and reusability by moving repeated lines of code into separate functions. Examples of such repeated tasks include increasing the size of pools and lanes, as well as creating new BPMN elements with a given set of attributes. The section that displayed the source code of the fixed BPMN file was also removed.

In addition, a further two guidelines were implemented as planned, namely Unlabeled Gateways and Absence of Existing Gateways, bringing the total number of implemented guidelines up to seven. All seven guidelines were then extended in order to support horizontal models with one pool and multiple lanes, as opposed to only single-lane models in the previous iteration.

4.2.4 Evaluation

The results of the second iteration were presented to Vostura, and the new version of the tool was generally well-received. The first point of discussion was the type of gateways to be added when fixing violations of the guideline *Absence of Explicit Gateways*. The company did not share my opinion that parallel gateways should be added. Exclusive gateways were preferred, not only because they were already the default gateways in Solution Manager, but also because of the possibility of adding labels to further guide the user to correctly update and adjust the automated corrections. Vostura also considered the prevalence of exclusive gateways relative to parallel gateways as a pro rather than a con.

Another point of discussion was about the labels automatically added during correction. The labels were sometimes easy to overlook upon import into Solution Manager. In addition, the added labels were sometimes positioned unfavorably, overlapping with other elements such as sequence flows. As SolMan had some capability to position labels, it was suggested to investigate if it was possible to adjust the layout of the inserted labels.

The final point of discussion was regarding next steps for the next iteration of the tool. One option was the implementation of additional guidelines. Another option was suggested by the company, namely adding configurability to the tool. All guidelines were being examined, and the methods to correct violations of the guidelines were set based on the decisions I had made during implementation. Adding configuration options to the tool would allow users to select which guidelines to check, or how the tool would correct the model. For example, users could choose the type of gateway to be created if one needed to be inserted into the model. As we did not come to an agreement on which option should be taken, I decided to consult with my thesis advisor. After listening to the two options, he convinced me that neither option was ideal. On one hand, the implementation of further guidelines would not add anything new to our understanding of the tool; on the other hand, adding configurability may be a case of adding a feature that is either currently or completely unnecessary. Instead, he proposed a third option, which was to conduct usability tests to determine how potential users of the tool would interact with the application, and what unseen issues could arise. The instructions provided, the layout, and the form of the report could also be evaluated. In addition, a usability test may help determine whether or not configurability is necessary. Following this conversation, I decided to conduct usability tests.

4.3 Iteration 3: Design Polishing and First Usability Tests

4.3.1 Goals

One goal of the third iteration was polishing the tool in preparation for the usability tests. Since not only the functionality but also the form of the app were going to be evaluated, it was important to make sure that both the visual design and the text included in the app were as clean as possible.

Another goal was to change the corrections performed under the guideline *Absence of Explicit Gateways* in accordance to the feedback given by Vostura during the

evaluation of the second iteration. This included not only switching from parallel gateways to exclusive gateways, but also inserting labels where necessary.

In addition, I was also going to investigate how gateway and sequence flow labels are positioned in BPMN. Once I figured this out, I could then determine how to set the layout of labels inserted during the correction of a BPMN model.

Finally, with the company's suggestion to include configurability into the tool, I was going to prepare the code for the possible introduction of this feature by creating separate functions for each guideline. This would also be good practice for any additional guidelines that would be implemented in the future.

4.3.2 Challenges

One challenge was determining what form labels inserted into the diagrams should take. Labels had to be obvious even when one was simply glancing at the fixed diagram. SolMan did not support changing colors within diagrams, so this was not an option. I decided that the created labels should be in all capital letters. This would more likely set it apart from other labels that the user would have created on their own, as well as from the default labels created in SolMan. In the end, I decided to use "GENERIC LABEL ADDED, PLEASE EDIT" for new sequence flow labels and "DEFAULT GATEWAY LABEL" for new gateway labels.

Related to this was the need to clarify whether the position of the labels could be adjusted. This was done by modeling and exporting models in both SAP Solution Manager and a third-party BPMN modeling tool, bpmn.io¹⁶. Then I examined the resulting source codes. I found that the positions of sequence flow labels in bpmn.io were set by adding a dc:Bounds node with the desired dimensions of the label. However, this node is ignored by SolMan, resulting in the labels being rendered in positions other than those set in bpmn.io. Indeed, when a BPMN file from bpmn.io is imported then exported in SolMan, the resulting file no longer includes the label's dc:Bounds node.

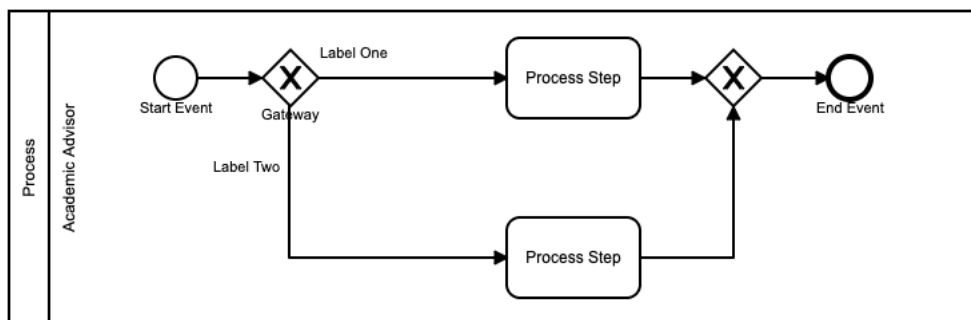


Figure 18: BPMN diagram with user-defined positions of sequence flow labels in bpmn.io

There are also options to set the placement of a label in SolMan. For example, one could set the position of a sequence flow's label near the source element, near the target element, or at the middle, as well as on which side of the sequence flow it

¹⁶<https://bpmn.io/> [Accessed: 2019-08-15]

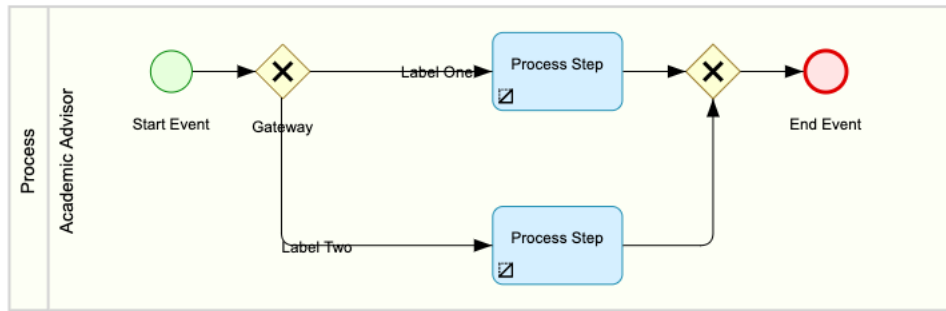


Figure 19: Same BPMN diagram as Figure 18 as imported into SAP Solution Manager

would be on. This means that one has a choice of six possible positions for a sequence flow's label. However, even when using this functionality, SolMan still does not export the position as a `dc:Bounds` node in the BPMN file. As a result, when exporting then importing a BPMN diagram in SolMan without changes, any adjustments that were made to the position of the sequence flow labels before exporting the model are lost. Thus, as there was currently no way to set the position of a label in a BPMN file such that SolMan interprets it correctly, the idea of positioning labels added by the validation and correction tool was dropped.

4.3.3 Results

The look and feel of the app was polished. Text in the app was also fixed. This included revising the instructions to use the tool as well as the report. For the report, short descriptions of each guideline checked as well as the rationales for including said guidelines were added. In addition, the text of the validation results was color-coded. If a diagram passed a test, the text regarding that guideline was displayed in green, while if any violations were found, then the text would be displayed in red.

When correcting violations of the guideline *Absence of Explicit Gateways*, exclusive gateways are now added instead of parallel gateways. As a result, labels for splitting exclusive gateways and their sequence flows also had to be added.

Finally, in order to facilitate the possible future implementation of configurability in the tool, guidelines were split off from the main validation and correction function into individual functions. Furthermore, each guideline had two functions, one for validation, and another for correction of violations of the guideline. This opened up the possibility of implementing alternate means of correcting a guideline violation.

4.3.4 Usability Tests and Evaluation

Initially, it was planned that employees of Vostura's clients who were tasked with modeling their business processes with BPMN in SAP Solution Manager would be asked to participate in the usability tests. Unfortunately, this was not feasible. First, at the time, there were only a handful of clients working with BPMN. Second, it was unrealistic to expect a company to set aside manpower on relative short notice for the tests. Third, the process of recruiting testers would have had to go through

4. Implementation and Evaluation

multiple levels of approval and communication, both internally as well as between the companies involved.

Instead, this first round of usability tests was performed with other Vostura employees. One was doing an internship at the company, another was a working student, and the third was a consultant who had done quality assurance on BPMN diagrams created by one of the company's clients. None of the three participants had seen or used the tool previously, so this was the first time they were using it. Although this was not the intended target audience of the product, this was a good compromise for the usability tests, as the participants could put themselves into the position of a QA analyst and of an employee of a client.

First, the participants were given a description of the tool then asked to rate their knowledge of BPMN, with 0 being no knowledge and 3 being advanced knowledge. After assessing a BPMN diagram to identify potential errors in it, they were asked to use the tool and import the resulting BPMN file into SolMan. Questions about general impressions of the app as well as how they felt about specific aspects of the tool were asked. A second diagram was assessed by the participants and the process repeated. Finally, participants were asked to give their final comments and suggestions on the tool. The guide questions as well as the diagrams used can be found in the [Appendix](#).

All three participants rated their knowledge of BPMN with the highest rating, 3, and were able to identify all the errors in the test models before using the tool.

There were three major points raised during the usability tests. The first issue, which was raised by all three participants, was that the instructions on the first page of the application were too long. They were in the form of a block of text describing how to use the tool and what behavior to expect from it. Two testers skimmed through the instructions and went straight into using the tool, while the third read through them carefully, later saying that they required too much concentration. Two of the participants then suggested using bullet points to present the instructions.

The second issue was that there was no way to look at the diagrams within the app. During the tests, the participants had a printed copy of the original BPMN diagram, and in order to look at the corrected diagram, they had to import the file into SolMan. This made it difficult to pinpoint what and where the corrections were made. All three participants expressed wanting a graphic representation of the diagrams inside the tool.

The third issue was regarding the labels created and inserted into the diagram. All three thought that they were sometimes either not recognizable or difficult to read. They suggested changing the positions of the labels. After the tests, I informed them that this was unfortunately not possible.

Apart from those points, the participants gave mainly positive comments regarding the tool. All three found it simple and intuitive to use, and all used it as expected, performing no unexpected actions. They were also mostly positive towards the generated report. The color-coding of the message about whether or not a guideline was violated was appreciated. The participants, however, did not necessarily agree on the level of detail provided in the report. One thought that the detail provided was good, while another thought it was too much.

The results of the third iteration as well as the usability tests were presented to Vostura. No additional comments, requests, or requirements were raised.

4.4 Iteration 4: Model Visualization and Second Usability Tests

4.4.1 Goals

The goals of the fourth iteration were based on the results of the usability test. The first was finding a way to display BPMN diagrams in the tool. Another was editing the instructions and the generated report. Finally, I wanted to finalize the app in terms of both form and functionality, perform defect tests, then run a final set of usability tests.

4.4.2 Challenges

The primary challenge during this iteration was integrating the rendering toolkit from bpmn-js [16] into the application. As SAPUI5 does not allow importing and running remotely-hosted third-party modules, a workaround needed to be used. I found instructions on how to include third-party scripts in SAPUI5¹⁷ and successfully imported a test script.

When attempting to integrate the bpmn-js toolkit into the tool during development of the proof-of-principle prototype, the main difficulty I had was trying to integrate the complete library consisting of multiple files and folders. There were no tutorials regarding this, so the idea was abandoned. A more careful perusal of the toolkit's page showed that there was a possibility to use and import a single file¹⁸, which could be integrated using the same method explained in the tutorial. This was successful, and BPMN diagrams could be rendered and displayed in the app.

Once that was completed, other issues related to displaying BPMN diagrams arose. The biggest challenge was trying to fit imported diagrams into the space provided for them in the app. Despite following online instructions and code that was supposed to produce this behavior, diagrams were still being displayed in a particular size and orientation. For example, parts of the diagram would be cut off because it was zoomed in too close, or only the lower half of the diagram was displayed. Instructions regarding how to zoom in and out of the diagram as well as how to drag it into position were added as a means to mitigate potential user issues related to this.

Another issue was that "X" marks within exclusive gateways were not being rendered and displayed despite being visible in SolMan. This was due to an attribute not being exported by SolMan into the BPMN file¹⁹. I added a quick fix so that the marker would be displayed.

Lastly, defect tests were performed in order to investigate if the guidelines were being checked and corrected according to expectations. Various diagrams were modeled, either adhering to or violating each guideline, for both single-pool and multi-pool horizontal models. While most of the test results were as expected, some defects were found. For example, when an end event had multiple incoming sequence flows, this did not show up as a violation of the Absence of Explicit Gateways guideline.

¹⁷<https://blogs.sap.com/2017/04/30/how-to-include-third-party-libraries-modules-in-sapui5/> [Accessed: 2019-08-15]

¹⁸<https://github.com/bpmn-io/bpmn-js-examples/tree/master/pre-packaged> [Accessed: 2019-08-15]

¹⁹<https://forum.bpmn.io/t/x-symbol-for-xor-gateway-in-viewer/2342/3> [Accessed: 2019-08-16]

4. Implementation and Evaluation

This was due to a programming error where end event nodes were not checked at all. Another defect found was that not all new splitting gateways as fixes to the same guideline were being created. This was due to a programming error where a variable name was used twice, thus inadvertently skipping past some of the loop indices. These and other errors were fixed, and the related test cases were retested, until all defects found were resolved.

4.4.3 Results

The bpmn-js toolkit was successfully integrated into the app. This allowed both the original and the modified diagrams to be displayed.

The instructions provided in the app were rewritten as a set of bullet points. The report was edited to reduce the amount of text displayed when describing the corrections applied to the model.

The layout of the tool was fixed. Potential layouts were presented to Vostura and one was chosen. Once validation and correction were completed, the page would display action buttons at the top, followed by the original BPMN diagram, then the diagram with corrections, and finally the report.

Some additional suggestions from the usability tests were implemented, such as adding more space between guideline sections in the report. In addition, defect tests were performed, and errors causing defects found during the tests were fixed.

4.4.4 Usability Tests and Evaluation

A second round of usability tests was performed with the updated tool, this time with a different group of testers. Three of the four new testers were fellow Bachelor Computer Science students, while the fourth was a Master of Education student whose subjects were Computer Science and Physics. None of them had seen or used a previous version of the tool. While the participants were chosen primarily because of the ease of recruitment, this would also be a good approximation of how people with little to no background in BPMN modeling would use the tool, in comparison to those with an advanced knowledge of BPMN in the previous round of tests.

The tests were run similarly to the previous usability tests. A side goal of the second round of usability test was to see if there was an associated learning effect to using the application, so this time participants had to assess four diagrams instead of two.

All four participants indicated that they had zero prior knowledge of BPMN. Thus, before they used and tested the tool, they were given a brief introduction to BPMN and the graphic elements used. They were also shown a syntactically correct BPMN diagram, Figure 1, as an example.

Most of the participants thought the instructions were clear and concise, and one commended the use of bullet points as opposed to a block of text. However, one participant mentioned that they still did not understand what the tool did and what steps needed to be taken. In addition, some suggestions were made to improve the instructions, such as adding the icons used in the app's buttons to the text.

The participants liked that the original diagram was displayed upon uploading a BPMN file, although one participant thought that this was the corrected version and

did not realize that they had to click on the "Validate and Correct" button in order to review and fix it.

Issues arose when the diagram was validated and both the corrected diagram and the report were displayed along with the original diagram. The primary issue was navigation. Trying to scroll from the original diagram to the fixed diagram then the report and back proved to be challenging, because depending on where the cursor was on the app, it was possible that only one of the diagrams would scroll and not the entire page. As a result, one participant ended up scrolling away one of the diagrams and had to start over in order to see it again. It was thus suggested to add borders to the sections where the diagrams were displayed. In addition, two scrollbars were being displayed on the page, and only one was actually useful, adding to the confusion of one participant.

For some of the participants, there was a bit of a learning curve regarding zooming in and out of the diagrams, however all of them eventually managed to do it. One suggested adding buttons that can be used to zoom in and out. There was also conflicting feedback regarding the instructions on how to use the zoom function. Whereas one criticized the instructions as being unclear, another commended the same instructions as being clear.

bpmn-js requires the display of their green logo when using their rendering toolkit. This became another source of confusion for a majority of the testers. One thought that this provided additional functionality to the tool. Two of the participants eventually clicked on the icon, which displayed information that bpmn-js was being used to render the diagrams.

Some expected to be able to make edits directly within the displayed diagram. One had tried to double-click on a generated label to edit it, and later explained that seeing "PLEASE EDIT" on it made them think that it was editable. In addition, all four testers suggested adding a clear way to display where corrections were made on the new diagram; indeed, one tester initially thought that the two diagrams were exactly the same. The use of color to highlight corrections was suggested.

While one participant specifically praised the order in which the elements were displayed, with the original diagram displayed at the top, followed by the fixed diagram, and then the report at the bottom, two other participants did not seem to notice that a report had been generated, and I had to prompt them to scroll down in order to view it.

All four participants thought that the report and the explanations provided were good, with additional compliments for the color-coded messages on whether or not a guideline had been violated. Most of the testers read through the reports carefully, although for some, I noticed that the amount of time spent reading decreased with each successive model that they assessed. One suggested only displaying guidelines that were violated in order to motivate users to read the report.

While two of the testers appeared to immediately understand that pressing the button labeled "Reset" would discard the current model and bring them back to the application's start page, one expressed uncertainty about what would happen when the button would be pressed, while another thought that it meant that changes made to the current model would be rolled back and the original model would remain.

In the end, all testers said that the tool was comfortable to use, and one said that

4. Implementation and Evaluation

they could imagine the usefulness of such a tool when dealing with a high number of large and complex models.

In terms of the tool having a potential learning effect, there was little to no real improvement in the performance of the testers as far as assessing the models is concerned. Still, the assessment performance of two participants remarkably improved between the first model and the second model. This could be because they read the first report and learned what was being checked by the model. On the whole, however, if a tester missed violations of a particular guideline in the first two diagrams, they would continue to miss them in the next two diagrams. In addition, none of the four participants were able to pinpoint labeled joining exclusive gateways. This may be because this was being checked and reported as part of the guideline Unlabeled Gateways, thus leading this particular rule to be overlooked.

As a result of suggestions made, feedback obtained, and behaviors observed during the usability tests, I made some adjustments to the tool. First, the text in the application was edited to make it more precise. For instance, a brief explanation of the "Reset" button was added, and the instructions for zooming in and out of a diagram were made more specific. Second, the design was slightly modified in order to make navigation and scrolling easier. For one, the outer scrollbar that had confused one of the participants was removed. The widths of the diagrams were also slightly reduced in order to increase the space on the app where one can scroll from the diagrams to the report and back again. The areas where the diagrams are displayed were also given a different background color as well as a border as suggested in order to distinguish them from the rest of the app. Third, the tool now automatically scrolls to the revised diagram once the validation and correction is completed, with the idea that perhaps more of the report is revealed on the screen, prompting users to scroll down further in order to view the report. Fourth, the text "PLEASE EDIT" in the inserted labels was replaced with "EDIT IN SOLMAN" to reduce the possibility that users would think they can edit the diagrams directly within the tool. Lastly, the guideline Unlabeled Gateways was split into its component rules in the report with two new headers, Unlabeled Splitting XOR Gateways and Labeled Joining XOR Gateways. This was to highlight the latter which participants did not seem to be aware of during their assessment of the models.

Some suggestions were not taken because they were not possible to implement. For example, although the green bpmn.io logo was mentioned by many participants as confusing, I am not allowed to remove it²⁰. Also, there does not seem to be a means within SAPUI5 of inserting icons inline with the text, so I could not implement that suggestion, either. Other suggestions were possible to implement but required greater effort than the changes already made. For example, it is possible to do overlays in bpmn-js which would allow me to mark changes in the corrected model. This and a few other examples will be discussed further in Section 5.2, Suggested Future Work.

The results of the fourth iteration as well as the usability tests were presented to Vostura. There were comments about making the layout of the start screen of the tool more SAP-like, but it was agreed to leave it as it is until a future version of the tool.

²⁰<https://bpmn.io/license/> [Accessed: 2019-08-16]

4.4 Iteration 4: Model Visualization and Second Usability Tests

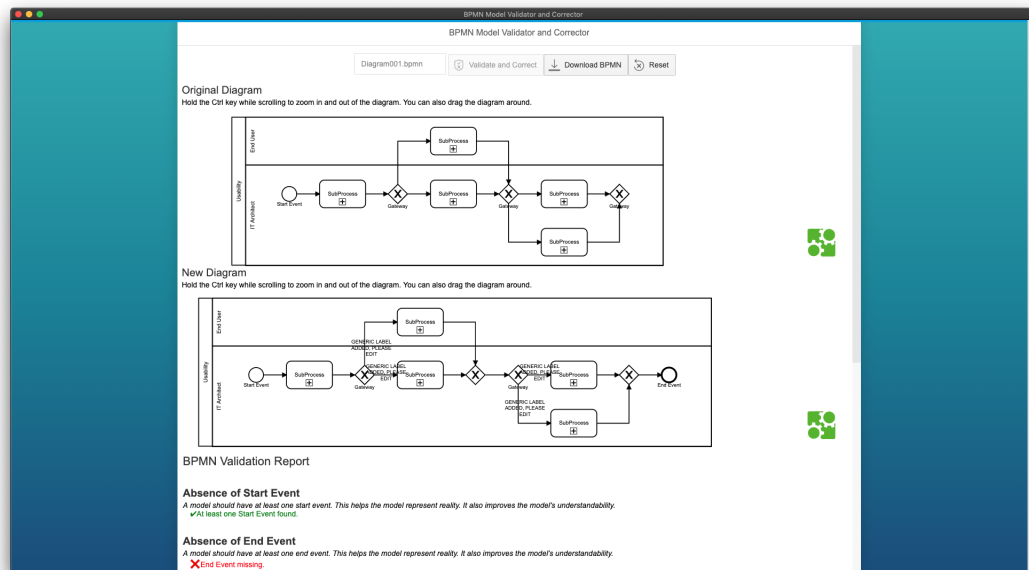


Figure 20: Tool after "Validate and Correct" button is selected, before the second round of usability tests

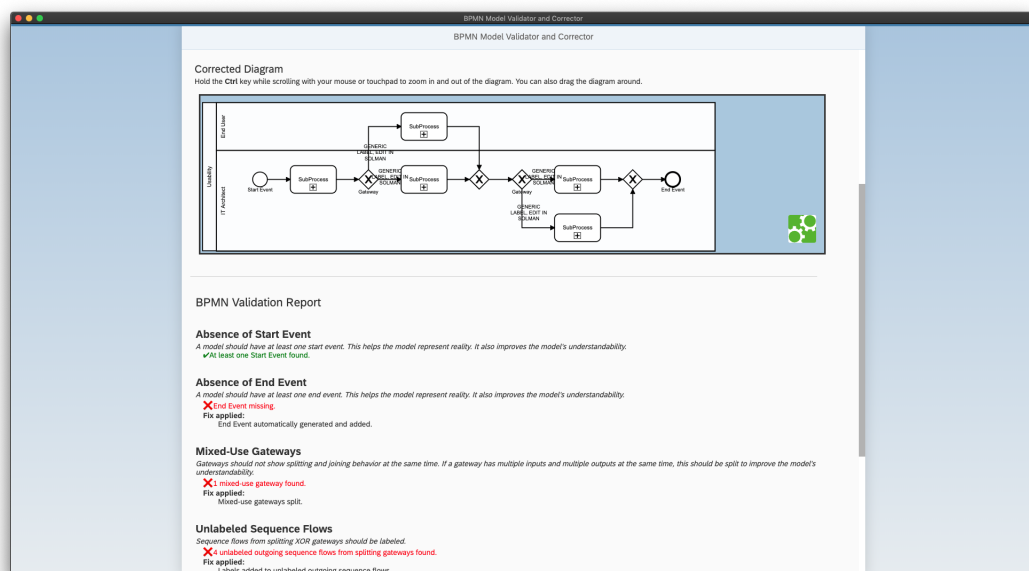


Figure 21: Tool after "Validate and Correct" button is selected, after the second round usability tests; note that the app now scrolls down to the corrected diagram

5 Conclusions

5.1 Summary of Results

The goal of this thesis was to implement a tool to improve the readability and usability of BPMN diagrams created in SAP Solution Manager (SolMan) by reviewing models based on a selection of guidelines and correcting any violations to them. The idea for the product came from Vostura GmbH, an SAP consulting company that focuses on Application Lifecycle Management (ALM) and SAP Solution Manager (SolMan). I started by going through related literature to determine what guidelines currently exist for the improvement of the syntactic and pragmatic quality of BPMN models. Then I formulated selection criteria in order to determine which of the pre-existing guidelines would be implemented in the tool. After using the criteria to select a set of guidelines, I determined that the tool would take the form of a web application built upon the SAPUI5 framework. I also decided that implementing the validation guidelines from scratch was more advantageous than integrating an existing validation library. This would provide flexibility in terms of dealing with peculiarities of working with BPMN files exported from SolMan, as well as with any future changes that would be made to it. I started development of the application with a proof-of-principle prototype which allowed me to see if my implementation plans were feasible. Upon successfully implementing the basic functionality of the app, I proceeded with an iterative and incremental development of the tool. Working in iterations allowed me to assess the product frequently, review my progress and plan the next steps with the product owners from Vostura, and adapt to various challenges and changes. Two usability tests were run during the product's development, one with employees of Vostura who have an advanced knowledge of BPMN, and another with students of Freie Universität Berlin who had zero knowledge of BPMN. Defect tests were also conducted to find and fix errors in programming that lead to defects.

In the end, I successfully developed the proposed tool for the validation and correction of BPMN models. While it is by no means a complete product, it is a fully functional prototype that contains the core functionality expected from a full application. The tool can validate and correct horizontal models with one pool and one or multiple lanes based on seven guidelines: Absence of Start Events (no start events in the model), Absence of End Events (no end events in the model), Unlabeled Sequence Flows (sequence flows from splitting XOR gateways have no labels), Mixed-use Gateways (gateways display both splitting and joining behavior), Unlabeled Gateways (splitting XOR gateways have no labels, while joining XOR gateways have labels), Unlabeled Events (events have no labels), and Absence of Explicit Gateways (non-gateway elements display splitting or joining behavior). The application displays the original diagram when a BPMN file is uploaded, and upon clicking the "Validate and Correct" button, the diagram with all the corrections is displayed, and a report detailing whether or not the guidelines were violated and what corrections were performed is generated and displayed. One can then download the modified BPMN file so it can be imported into SolMan, and start the process over with a new file.

5.2 Suggested Future Work

As I mentioned earlier, the BPMN validation and correction tool is not necessarily a complete product. In this section, I discuss some additional features that can be implemented as well as next steps that can be performed in order to turn this working prototype into a market-ready product.

- **More guidelines and wider scope.** I initially considered a total of 61 different guidelines, as mentioned in Section 3.1. Seven were implemented for this thesis. This is lower than the current lowest level of support provided by a commercial BPMN modeling tool, as discussed in [Related Work](#), so adding more guidelines would be a good idea, even if not every guideline's violations can be easily or automatically fixed. In addition, algorithms to analyze text can be integrated in order to implement guidelines that deal with content, such as labels on splitting exclusive gateways being in the form of a question or interrogative phrase. The tool also currently only works with horizontal models with one pool. Business processes can also be modeled vertically, and while most of the current validation checks and some of the corrections, such as those regarding missing labels, work with vertical models, some of the correction algorithms, in particular those where new elements must be added into the model, need to be adjusted for such models. Moreover, it is possible to create models with multiple pools in SolMan. The current validation and correction functions would also have to be adjusted for such cases.
- **Configurability.** As discussed in Section 4.2.4, the product owner Vostura suggested adding the ability to allow users to configure the tool. One thing to configure would be how a particular violation of a guideline would be corrected. For example, if a start event is missing, then the user could choose how to determine which is the first element of the process model; or, the user could decide what kind of gateways would be added when the guideline Absence of Explicit Gateways is violated. Another option suggested by the company would be the ability to ignore certain guidelines when reviewing or fixing a model. Also, one participant in the second round of usability tests suggested hiding guidelines with no violations in the report, which could be included as another option that users can configure themselves. The goal of configurability would be to allow users more control over their experience and over the corrections performed, as at the moment, they are limited by what I suggest as the best means to correct the problems.
- **More obvious indication of changes to diagrams.** As mentioned in Section 4.4.4, the participants of the second usability test suggested making it clearer what and where the corrections performed are on the corrected diagram. One possibility is adding annotations to the displayed diagram with corrections. bpmn-js supports the use of HTML overlays to display notes and colored annotations to a BPMN diagram²¹. This could be used to draw the user's attention to changes in the corrected diagram.

²¹<https://github.com/bpmn-io/bpmn-js-examples/tree/master/overlays> [Accessed: 2019-08-23]

- **Add zoom and reposition buttons to diagram displays.** Another suggested feature from the second round of usability tests is the addition of buttons to be able to zoom in and out as well as reposition a displayed diagram. This came from comments and observations that there were difficulties zooming in and out of the BPMN diagrams displayed in the tool. This feature already exists in the SolMan modeling tool, and it would be helpful to add this to the BPMN validation and correction tool. There is an open source example of how to implement this²², but time needs to be taken in order to properly implement and test this feature.
- **Allow direct edits to the corrected diagram.** One participant in the second usability test tried editing the diagram directly from within the application. While this was due to a misunderstanding of the generated labels, this is a valid feature to add to the tool. While implementing a full-featured modeling tool is not the goal of this product, allowing the user to immediately make some changes, such as specifying the labels to be added, could be a good idea.
- **Proper integration with SAP Solution Manager.** In Section 1.2, I mentioned that "integration" in this thesis meant not having to download special software or use a non-SAP web application for the purposes of model validation, and that this tool can be opened directly from within SAP. As of this time, the process of being able to open the app directly from SAP is being undertaken by Vostura. However, more interesting and challenging would be a better, more direct integration with SAP Solution Manager. Currently, users would have to export a model as a BPMN file from SolMan, upload this file into the app, download the corrected file, then import the file with corrections back into SolMan. The goal is to be able to review BPMN models without the need to export them but to open them directly from within the app, and to be able to save these changes without having to download and import the file. There is an API which allows one to access models in SolMan, but this returns a JSON²³ file rather than an XML file²⁴. A parser to transform the JSON into the same XML that the SolMan exporter would create and then back again would be needed. In addition, the actual means of calling the API does not seem to be documented online.
- **More usability tests.** The usability tests performed as part of this thesis were good compromises given the time and availability constraints. However, this does not paint a complete picture of how the average user would use the tool. One example is how the test participants interacted with the validation and correction report. On one hand, the Vostura employees from the first round of testing may have been so knowledgeable in BPMN that they did not need to thoroughly read the report. On the other hand, the Freie Universität Berlin students from the second round of testing may have had so little knowledge in BPMN that they had to spend more time reading the report. Those expected

²²<https://github.com/prayerslayer/bpmnio-zoom-control-example> [Accessed: 2019-08-23]

²³JavaScript Object Notation, a human-readable file format for transmitting data objects as attribute-value pairs.

²⁴<https://wiki.scn.sap.com/wiki/display/SM/Files> [Accessed: 2019-08-23]

5. Conclusions

to use the product likely have a BPMN familiarity level somewhere in between. Also, as one participant in the second round of tests mentioned, it is possible that some users may not be interested in the report, only in the promise that the resulting diagram has been corrected of its various issues. The diagrams used in the usability tests would also have to be improved. I discovered that using generic diagrams, where the labels used had no real semantic meaning, was confusing to both groups. Diagrams should be used that more closely resemble those that would be made by users of the application. Finally, it would be interesting to examine potential learning effects more closely, if it is possible to improve one's modeling practices through the use of this app.

5.3 Personal Lessons

Using an iterative and incremental development process allowed me to assess myself more frequently. Because I gave myself time at the end of each iteration to step back and go over things, it gave me the opportunity to identify behaviors that I could improve for future iterations, and ultimately, for future projects. For example, when the first iteration started, I directly started programming almost blindly. I then had to keep reworking my code as I would miss important things while coding. I learned that taking time to think things through and describe the algorithm to be implemented on paper through pseudocode resulted in better code and surprisingly less time for implementation, as I was using my time more efficiently.

In terms of non-technical, non-programming skills, working on the thesis gave me a better insight on documentation. It was not just about adding comments to code, although that is also part of documentation. I kept a personal journal of decisions made and progress on the development of the tool, including small changes and fixes as well as problems I had faced. While this was helpful in writing my thesis paper, this was even more helpful when taking stock of what the application can and can not do, and helped in terms of providing information for the handover of the project to the company. Personally, it was also a good way of seeing that I was making progress, even if it felt that I was not doing enough at times, or things were not improving.

In addition, perhaps the biggest personal lesson working on this thesis and the application is regarding personal productivity and work ethic, particularly in difficult situations. While it was easy to keep programming and working when I was "in the flow", when I experienced roadblocks, such as a bug I could not seem to fix, trying to force my way past it did not work most of the time. Instead, what worked was taking a break, either focusing on some other aspect of the program or even stopping completely for the day, then returning to it at another time. For me personally, it is more unproductive to keep going when one is faced with mounting frustration. I then found ways apart from coding to be productive during these times, such as improving my documentation or performing additional tests.

Bibliography

- [1] OMG, “BPMN Specification - Business Process Model and Notation.” <http://www.bpmn.org/>. Accessed: 2019-04-22.
- [2] O. I. Lindland, G. Sindre, and A. Solvberg, “Understanding quality in conceptual modeling,” *IEEE software*, vol. 11, no. 2, pp. 42–49, 1994.
- [3] M. Geiger and G. Wirtz, “Bpmn 2.0 serialization-standard compliance issues and evaluation of modeling tools,” *Enterprise Modelling and Information Systems Architectures (EMISA 2013)*, 2013.
- [4] M. Geiger, P. Neugebauer, and A. Vorndran, “Automatic standard compliance assessment of bpmn 2.0 process models,” in *ZEUS*, pp. 4–10, 2017.
- [5] A. Correia and F. B. e Abreu, “Adding preciseness to bpmn models,” *Procedia Technology*, vol. 5, pp. 407–417, 2012.
- [6] J. De Bock and J. Claes, “The origin and evolution of syntax errors in simple sequence flow models in bpmn,” in *International Conference on Advanced Information Systems Engineering*, pp. 155–166, Springer, 2018.
- [7] M. Zur Muehlen and J. Recker, “How much language is enough? theoretical and practical use of the business process modeling notation,” in *Seminal Contributions to Information Systems Engineering*, pp. 429–443, Springer, 2013.
- [8] I. Moreno-Mones de Oca and M. Snoeck, “Pragmatic guidelines for business process modeling,” tech. rep., KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management, 2014. Available at SSRN 2592983: <https://ssrn.com/abstract=2592983>.
- [9] I. M.-M. de Oca, M. Snoeck, H. A. Reijers, and A. Rodríguez-Morffi, “A systematic literature review of studies on business process modeling quality,” *Information and Software Technology*, vol. 58, pp. 187–205, 2015.
- [10] M. Snoeck, I. M.-M. de Oca, T. Haegemans, B. Scheldeman, and T. Hoste, “Testing a selection of bpmn tools for their support of modelling guidelines,” in *IFIP Working Conference on The Practice of Enterprise Modeling*, pp. 111–125, Springer, 2015.
- [11] F. Corradini, F. Fornari, S. Gnesi, A. Polini, B. Re, and G. O. Spagnolo, “Quality assessment strategy: Applying business process modelling understandability guidelines,” 2017.
- [12] F. Corradini, A. Ferrari, F. Fornari, S. Gnesi, A. Polini, B. Re, and G. O. Spagnolo, “A guidelines framework for understandable bpmn models,” *Data & Knowledge Engineering*, vol. 113, pp. 129–154, 2018.

Bibliography

- [13] T. Rozman, G. Polancic, and R. V. Horvat, "Analysis of most common process modeling mistakes in bpmn process models," in *2008 BPM and Workflow Handbook*, University of Maribor Slovenia, 2008.
- [14] B. Silver, *BPMN Method and Style, with BPMN Implementer's Guide*. Cody-Cassidy Press, 2011.
- [15] I. Femia, "#HTML5, #openui5, #sapui5 and #sapfiori... How are they working together?," Oct 2015. Available at <https://blogs.sap.com/2015/10/14/html5-openui5-sapui5-and-sap-fiori-how-are-they-work-together/>. Accessed: 2019-08-21.
- [16] bpmn io, "bpmn-io/bpmn-js: A BPMN 2.0 rendering toolkit and web modeler." <https://github.com/bpmn-io/bpmn-js>. Accessed: 2019-08-01.
- [17] V. Farcic, "Software development models: Iterative and incremental development," Jan 2014. Available at <https://technologyconversations.com/2014/01/21/software-development-models-iterative-and-incremental-development/>. Accessed: 2019-08-22.
- [18] C. Larman, *Agile and iterative development: a manager's guide*. Addison-Wesley Professional, 2004.
- [19] W. E. Deming, *The new economics for industry, government, education*. MIT press, 2018.

A Appendix

A.1 Questions of the Usability Tests

What follows are the guide questions I used when performing the usability tests. While I allowed for spontaneous questions based on comments and behavior during the tests, these questions were mainly for my use to remind myself to ask them as well as to make sure my questions were more open-ended and not simply answerable by "yes" or "no".

As the usability tests were performed in German, both the English and German translations of the questions are provided here.

Question 5 was not asked during the second usability tests as the participants were unfamiliar with SAP Solution Manager.

1. How would you rate your proficiency with BPMN? / *Wie bewerten Sie Ihre BPMN-Kenntnisse?*
 - 0 - no proficiency / *keine BPMN-Kenntnis*
 - 1 - low proficiency / *niedrige BPMN-Kenntnis*
 - 2 - medium proficiency / *mittlere BPMN-Kenntnis*
 - 3 - high proficiency / *fortgeschrittene BPMN-Kenntnis*
2. How are the instructions? (Are they clear/easy to understand/confusing?) / *Wie finden Sie die Anleitung? (Ist sie klar/einfach zu verstehen/verwirrend?)*
3. Check the following BPMN diagram. What would you say was wrong with the model? / *Überprüfen Sie das folgende BPMN Diagramm. Was ist falsch an dem Modell?*
4. Run the tool to validate and correct the model. How do you find the descriptions of the inspected guidelines? (Are they clear/easy to understand/confusing?) Are the displayed errors in the model what you expected? How do you find the actions taken to fix the model? (Are they reasonable?) / *Führen Sie das Tool aus, um das Modell zu überprüfen und korrigieren. Wie finden Sie die Beschreibungen der geprüften Richtlinien? (Sind sie klar/einfach zu verstehen/verwirrend?) Entsprechen die angezeigten Fehler im Modell Ihren Erwartungen? Wie finden Sie die Maßnahmen zur Korrektur des Modells? (Sind sie nachvollziehbar?)*
5. Import the corrected model to SAP Solution Manager. How do you find the corrected diagram? / *Importieren Sie das korrigierte Modell in SAP Solution Manager. Wie finden Sie das korrigierte Diagramm?*
6. How do you find the tool? What do you like about it? What does the tool do well? What can be improved? Are there any features you would like to see in the tool? / *Wie finden Sie das Werkzeug? Was mögen Sie daran? Was macht das Tool gut? Was kann noch verbessert werden? Gibt es Funktionen, die Sie im Tool sehen möchten?*

A.2 Models Used for the Usability Tests

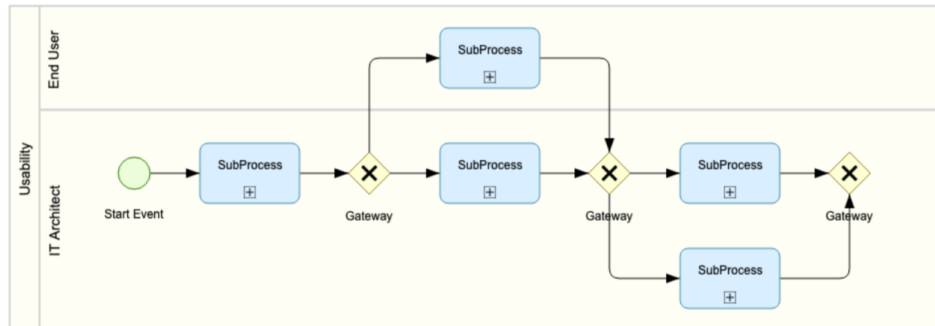


Figure 22: Usability test diagram 1

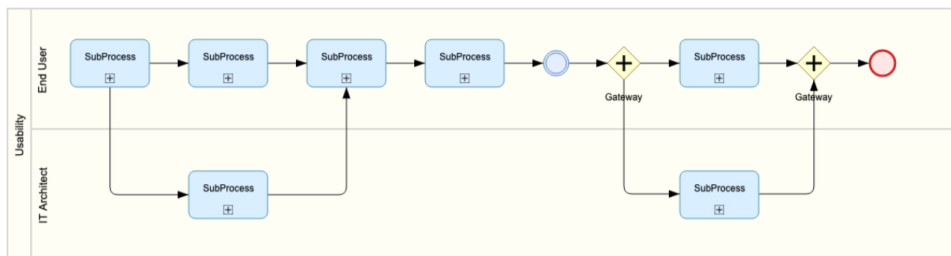


Figure 23: Usability test diagram 2

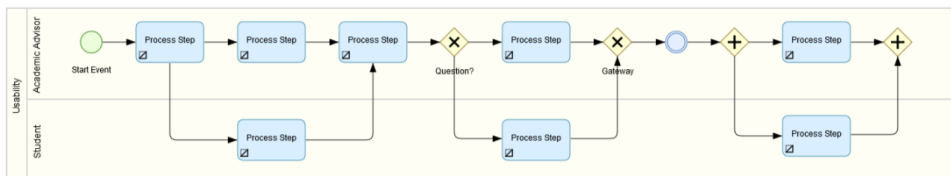


Figure 24: Usability test diagram 3

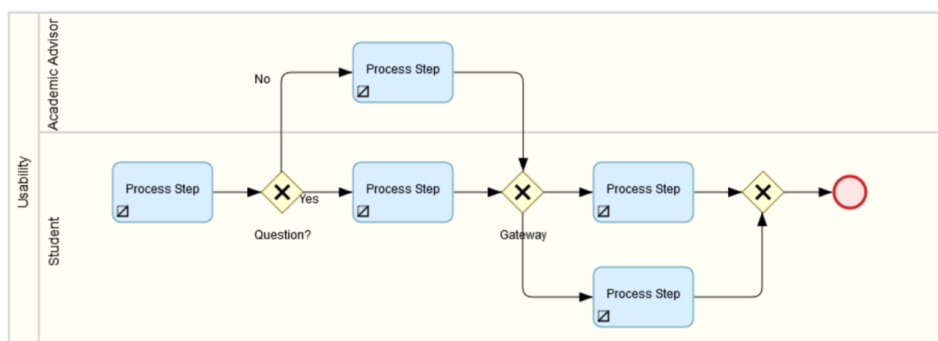


Figure 25: Usability test diagram 4