TARTU UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
INSTITUTE OF COMPUTER SCIENCE

Svitlana Vakulenko

Extraction of Process Models from Business Process Descriptions

Master's Thesis (30 EAP)

Supervisor:

Marlon Dumas, University of Tartu

Author: . . . . . "." . . . . . " May 2011

Supervisor . . . . . "." . . . . . " May 2011

Professor . . . . . "." . . . . . " May 2011

TARTU, 2011

Acknowledgement

First of all, I want to thank Marlon Dumas, my supervisor, for his support of my “crazy idea”, which put this thesis into life. He was also the one, consulting me on the application of the business process modeling approach and general research methods. I acknowledge his great contribution to the development of the process extraction method proposed in this thesis.

I express special gratitude to Maarika Traat for her support and valuable comments, which helped in shaping the final version of this thesis.

Thanks to my parents and friends, who constantly reinforced my motivation, for their care and faith in me.

Table of Contents

List Of Abbreviations

CA – composite activity;

BP – Business Proces;

BPM – Business Proces Modelling;

BPMN – Business Proces Modelling Notation;

IDE – Interactive Development Environment;

IE – Information Extraction;

NL – Natural Language;

NLP – Natural Language Processing;

POS-tag – Part-Of-Speech tag;

UML – Unified Modelling Language.

1 Introduction

Modern organizations maintain valuable information about their business processes in their manuals of policies and procedures and other internal business documentation, but the understanding of how the processes work from such descriptions is difficult. Graphical process modeling is often used as a complement to textual descriptions in order to enhance understandability and to facilitate communication and knowledge sharing. But for companies which have not done a process diagramming effort, extracting process diagrams from their documentation is a painstaking effort.

Natural language processing (NLP) technology has advanced a lot in recent years. This has made it possible to apply NLP methods in order to automate text processing tasks in various fields. In particular, NLP methods have already been successfully applied for the purpose of extracting models from Use case specifications[6], [7], [15], [20].

The goal of the research presented in this thesis is to analyze opportunities to semi-automate process model extraction and to propose a method for transforming textual narratives into process models. Given a textual description of a process in natural language (English), the method is expected to generate a structured process model, which can be visually represented as a diagram.

The crux of the proposed method is a set of *patterns* designed to identify and to extract *process elements* (i.e. activities, actors, artifacts) and relationships between them (specifically control flow relations). The elements and relationships captured in this way are represented using a simple meta-model for structured process models. From the resulting process models, it becomes straightforward to generate diagrams in any suitable notation (e.g. in BPMN or UML Activity Diagrams).

To illustrate the scope and purpose of the proposed method, we consider a simple example of a university enrollment process. A textual narrative of this process is given in Figure 1.1, while the output of the proposed method (i.e. a corresponding process diagram) is presented in Figure 1.2.

> Student fills an online form. Student may attach supplementary documents to the form.
>
> Then the student submits the application electronically or sends it physically by post.
>
> When student service receives the application, it is checked for completeness. If some of the required documents are missing, the student should send the missing or incorrect documents.
>
> Then the students service sends the certified copies of the degrees to a specialized agency (ENIC). While ENIC verifies the validity of the degrees,the English language test results are checked online by an officer at the students service. If both checks are passed, student application is accepted. Otherwise the application is rejected.

Figure 1.1. Input: textual process description.

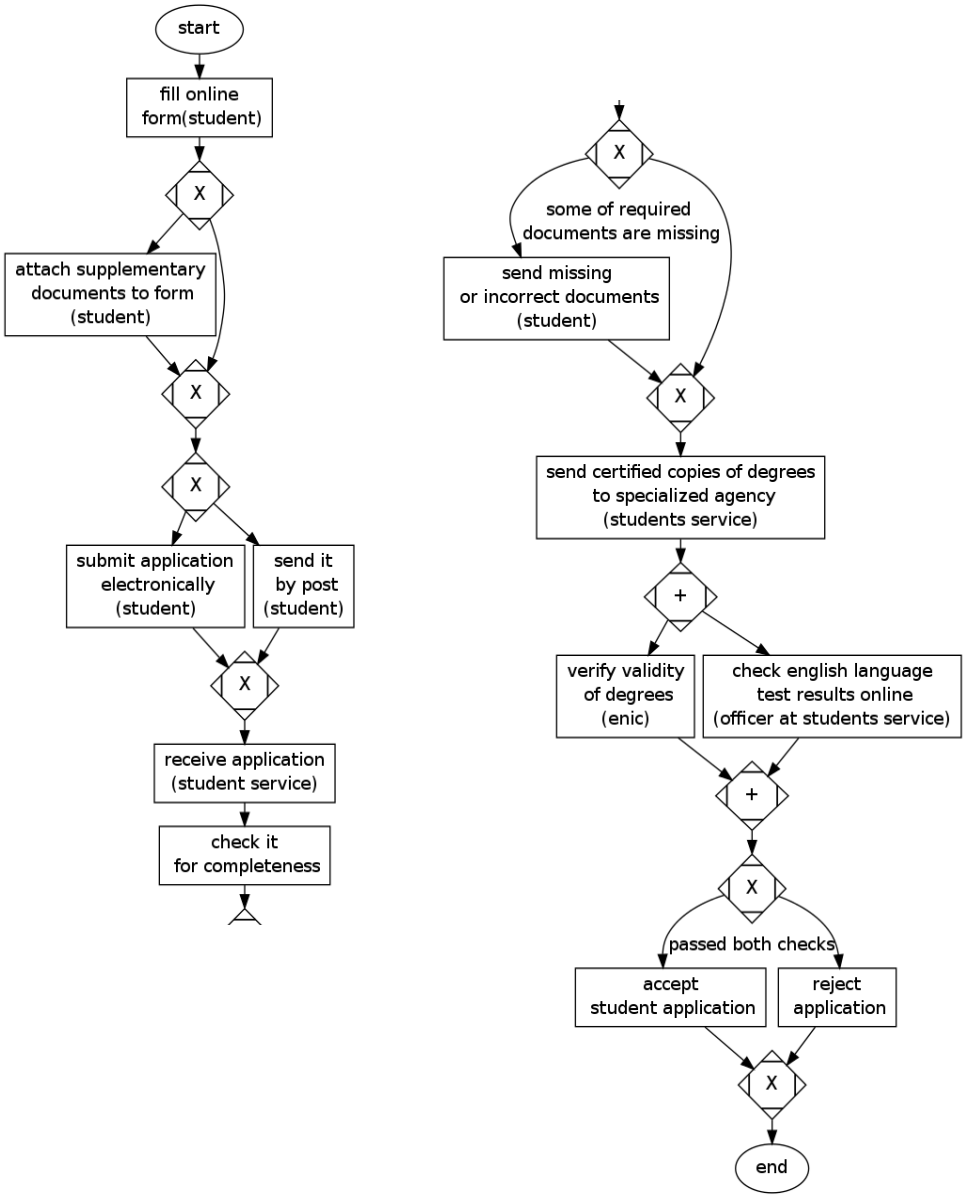


Fig.1.2. Output: process diagram.

The proposed method has been implemented as a proof-of-concept prototype that takes as input textual process descriptions and produces process diagrams that can be displayed using the GraphViz tool [11]. The diagram shown in Figure 1.2. was generated using the implemented prototype. Note that the diagram has been cut into two parts (left and right side of the figure) for presentation purposes.

The remainder of the thesis is structured as follows. Chapter 2 gives an overview of the state of the art in the area of automated (process) model extraction from textual documents. Chapter 3 describes the proposed process model extraction method. Chapter 4 presents the prototype implementation, while Chapter 5 discusses the validation of the prototype on the basis of real scenarios and discusses the limitations of the method and opportunities for improvement. Finally, Chapter 6 summarizes the contributions and outlines directions for future work.

2 State-of-the-art

The only publicly available software that has similar functional to the one envisaged in this thesis is Raven – a commercial tool by Raven-Flow, Inc. Raven implements an automated method of transforming a textual process description into a BPMN diagram. It can be accessed through a free trial via the RavenCloud on-line service [23].

Raven Cloud identifies actors (swimlines), artifacts, activities, entry and exit points, alternative and repetition control flow. However, Raven algorithm uses an artificial semi-natural language for its input. It is not explicitly reinforced by the program, but the text is parsed correctly only if it corresponds to a pattern from a set of predefined patterns, which is, in fact, extremely limited. Cross-validation of our prototype against Raven Cloud revealed that Raven Cloud fails to produce any meaningful model from the textual input, which our prototype is able to process correctly. Example of a process model cross-validated across our tool and Raven Cloud can be found in Appendix 5, 6.

A recent research, addressing the challenges of an automated process model extraction from business documents, is presented in [19]. The paper proposes a rapid approach to business process modeling - Rapid Business Process Development (RBPD), which can be also called process discovery. RBPD-tool takes as an input heterogeneous business documentation, parses it and extracts information, which might be related to a business process. The drawback of this approach is that it outputs disconnected pieces of process, not linked with each other – Figure 2.1. While it can be of help to an analyst to identify changes of the flow and other events, this approach is unable to provide a holistic view on the process. In contrast, our method assumes that an input text contains only process description, all information unrelated to the process has to be filtered out beforehand.

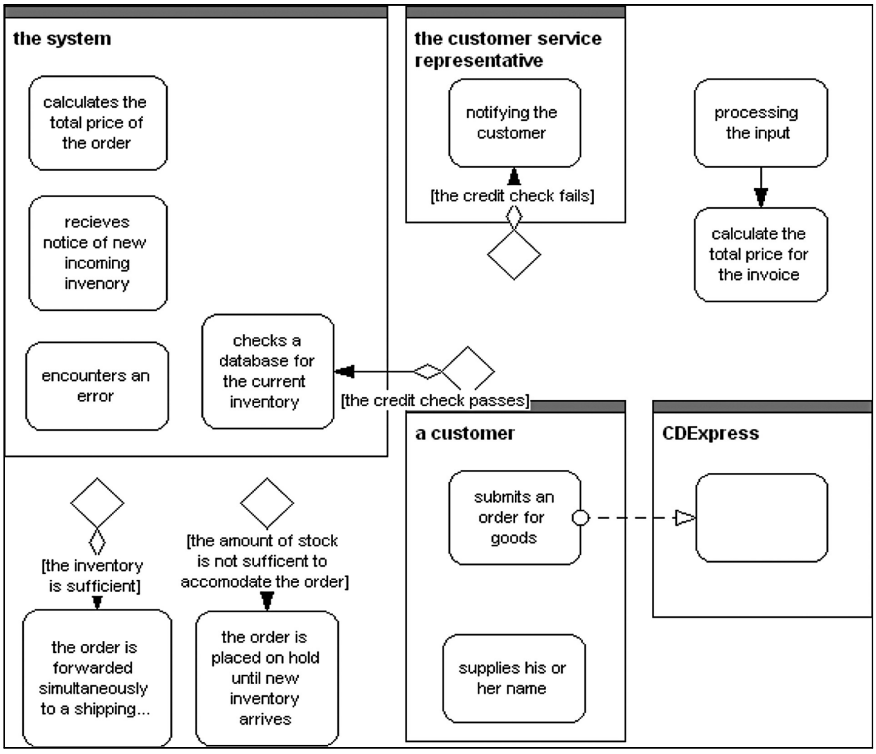


Figure 2.1. BPMN Proto-Process Model Fragments Extracted from Text by R-BPD tool [1].

A method for automated process extraction from plain text was also proposed in [16], [17], [18]. It was developed in a framework of an on-going project, as a part of a process elicitation approach through collaborative “story-telling”. The processes are to be extracted from “stories” - textual process descriptions, created by several users through GroupStoryTelling tool. The Story Mining method employs a similar to our approach. By now the method was reported successfully identifying only Atomic Activities(Tasks), but no Composite Activities (flow of actions in [9]). Chapter 3 of this thesis presents common patterns, including corresponding "signal words", for identifying all 4 types of Composite Activities(Control Flows) and, moreover, discusses the possible limitations for automating their extraction. The prototype tool, that was developed in our research, is already able to identify a subset of Composite Activities(CA).

Application of NLP techniques for automated model extraction from Use Cases and Requirements specifications has already an old tradition, dating back to [9]. Several methods were developed and implemented. Most of them focus on generating UML diagrams (class, activity, sequence, etc), for example[6], [7], [15], [20]. Some of the methods restrict the input language to a template-based one, while others are in support of a natural language usage for requirements descriptions.

While process elicitation from use case specifications employ similar information extraction methods to the one described in this thesis, the language of the use cases is, in general, simpler, more standardized and therefore easier to analyze than the one of business process(BP) descriptions. Typical use cases do not use as many various control flow structures(alternative, parallelism, repetition), as BP descriptions normally do. Use cases tend not to have parallelism because they generally describe interaction of one user with a computer system, and the user performs only one task at a time. In contrast, process models describe activities performed by multiple actors. Furthermore, when there are different ways of interacting with a system, these alternatives are generally described in different use cases. In contrast, if there are several methods of performing an activity, all of them will be captured in one process model.

3 Problem Analysis

3.1 Scope and method of the analysis.

A process is a system consisting of activities, which can be either atomic activities (also called tasks) or composite activities consisting of several other activities. Elements of a system, by definition, must be related to each other, forming an interconnected component – the system itself. Hence, all activities, related to the process, have to be connected to this process (meaning, at least to one of the activities in the process). Thus, process is a set of interconnected activities.

The main characteristic of a process as a system is that it is evolving in time, which reflects a certain sequence order between its elements. Hence, process is an ordered chain (sequence) of interconnected activities. The most basic relation between activities in a process is a simple sequence, meaning that activities are happening one at a time, with one activity following another. More complex relations between activities also exist (e.g. parallel execution, alternative execution), which make the process deviate from a pure sequence.

The same characteristics discussed above are found in the "story-telling" approach, which humans use to convey information – "to tell a story". In a traditional "story", by convention, facts (e.g. about events) are told in a logical sequential order. Narrative, which contains a "story" is a system with paragraphs sentences and words, as its elements. This system also has a certain structure: order of the paragraphs in a text, sentences in a paragraph, words in a sentences - imply the same sequential order on the context, which they contain.

A process description in natural language is a type of a "story", in a form of a text it gets the characteristics of a narrative. Here we draw a parallel between a process and its textual description – narrative. They both have elements ordered in a certain structure(s). Sequence is the basic and the most common type of order, as it reflects the category of time.

Thus, our method for translating a narrative into a process model is based on the assumption that the input narrative has a logical sequential order, which means that all activities are listed in a sequential order, unless another order is explicitly indicated by certain "signal words" which correspond to a change in a natural flow of the "story", which is a "baseline" corresponding to a sequence.

Despite huge advances in NLP in recent years, processing of a natural language still remains a challenge, as the system of natural language is much more complex in comparison with any computer-based notation. The main problem to deal with, when using NLP approach, is the ambiguity of natural language. In this respect some researches advocate the use of controlled natural language, which forces users to follow a set of strict guidelines, which help to avoid ambiguity [5]. Existing methods for automated model extraction from textual descriptions, such as [23], follow an approach based on controlled natural language. In other words, users are asked to follow strict conventions, while writing textual descriptions. However this approach is not suitable for analyzing texts, which already exist and are written in a free-form. As our research aims at generating models from already existing process descriptions, we have to develop methods for analyzing natural language text and define the ways to deal with its ambiguity.

The language typically used for textual business process descriptions is quite complicated

for understanding by human readers. It usually comprises long sentences with complex structures, abbreviations, special terms, etc. At the same time, we observe that textual process descriptions are full of "if..else" and other lexical constructions following certain patterns. They often look more like a program code, which facilitates the task of automatically extracting process elements and relations between them.

In order to extract a process model from a textual process description, it is often not necessary to know the meaning (semantics) of every word, but rather to understand the relations between them. These reltions can be inferred from the syntactic or morphological structure. Accordingly, our proposal is based on the premise of the existence a ***business process sub-language*** that is commonly used in textual process descriptions. This business process sub-language does not contain business terms, but rather a set of grammar-based patterns corresponding to process elements and their relations, including specific "signal words" indicating a change in the flow of control between activities in the process. Hence, we adopt a "best-effort" approach driven by a number of patterns corresponding to typical grammar structures found in business process documentation.

It has to be admitted that the proposed approach aims at identifying only block-structured process models, plus references (i.e. "go-to" statements) to previous elements in the process model [8]. Block-structured process models are models, that are composed of atomic tasks and four types of blocks: sequential blocks, parallel blocks, if-then-else blocks, repetition blocks. Each block (also called a composite activity) has one or several other blocks or atomic tasks as children.

Block-structured process models are easier to produce and they are also more comprehensible than unstructured process models [13]. In the research presented in [19], on the basis of two libraries of process models extracted from industrial practice, it has been found that between 54% and 65% of process models were entirely block-structured, while the remaining ones were block-structured except for, at most, one non-block-structured fragment per model. Also, various techniques exist for automatic conversion of unstructured process models into equivalent block-structured ones [3]. Although these automated transformation techniques do not work in 100% of the cases, they provide evidence to the fact that structured process models represent a large class of process models. Thus, the decision to aim research at process extraction from structured process models seems to be a reasonable trade-off towards reducing complexity of the system.

In order to implement a software-tool for automated process extraction from textual narratives, it is essential to analyze common patterns for identifying process elements in the text and formulate general rules according to these patterns. For this purpose a set of narratives, consisting of industrial process descriptions and academic case-studies was analyzed [4], [12], [14], [21][1]. The main assumption was that the set of case-studies, selected for the analysis, should contain patterns, which are common across business process descriptions in general.

Business process specifications are not standardized, however they often contain similar structures. Language style (vocabulary and lexical structures used) is normally consistent and repeated through all the document. The structure also depends on the size of the document. In

1 We also made use of two unpublished case studies: "Improving the 'IT Incident Management' process at FictOrg" and "Improving the 'Compulsory Third Party Insurance Claims' Process at New Generation Finance (NGF)", written by Jan Recker, Queensland University of Technology.

general, a compact description contains a concentrated text, which does not contain numerous details irrelevant to the process. Typical structure of a middle-sized BP description document is the following:

- Title - contains name of the process;
- Introduction – presents purpose of the document;
- Process Overview – describes process on a high level of abstraction;
- Process Description – presents the detailed description of the process itself;
- Exceptions and additional details about the process;
- Conclusion.

Hence, the part to be used for process model extraction is Process Description (sometimes Exceptions, if they are structured). Process Overview may contain process description on a more abstract level, hence Composite Activities and their relations can be extracted from there.

Text structure itself says a lot about sequence ordering and grouping of the Activities in the Process described. Activities, in most of the cases, are ordered sequentially in the text of a business process description (excluding separated sections alike Exceptions). Process Description can be structured into Sections which can denote composite activities described in Process Overview, e.g. “Incident resolution and workaround”, “Incident review and closure”, etc. Moreover, the order and grouping into paragraphs and sentences gives another clue for identification of composite activities.

The method used for the analysis of the case-studies is based on inductive reasoning and formalization. Each of the cases was carefully studied and a mental model of the process described in the text was constructed. After that, the text elements were analyzed and language structures corresponding to process elements and their relations, and "signal words" (signaling changes in the workflow) were inventoried and grouped. Out of each group of examples sharing common features, a general rule (pattern) was induced and formally captured. Thus, the human knowledge concerning the methods for process model extraction from a natural language text was translated into a set of generalized rules. Then, these rules can be translated into programming language for a software tool able to automatically extract process models.

The output process model is abstract and was not meant to follow any specific notation, thus it can be utilized to build diagrams in different notations. However, the process modeling notation used for illustration purposes throughout the thesis corresponds to a subset of the BPMN notation.

For the sake of analysis, the set of process elements for automated extraction was limited to the 5 basic classes:

- Activities;
- Artifacts;
- Actors;
- Entry&Exit Points;
- Control Flows.

Relations between these types of process elements are presented in a class diagram:

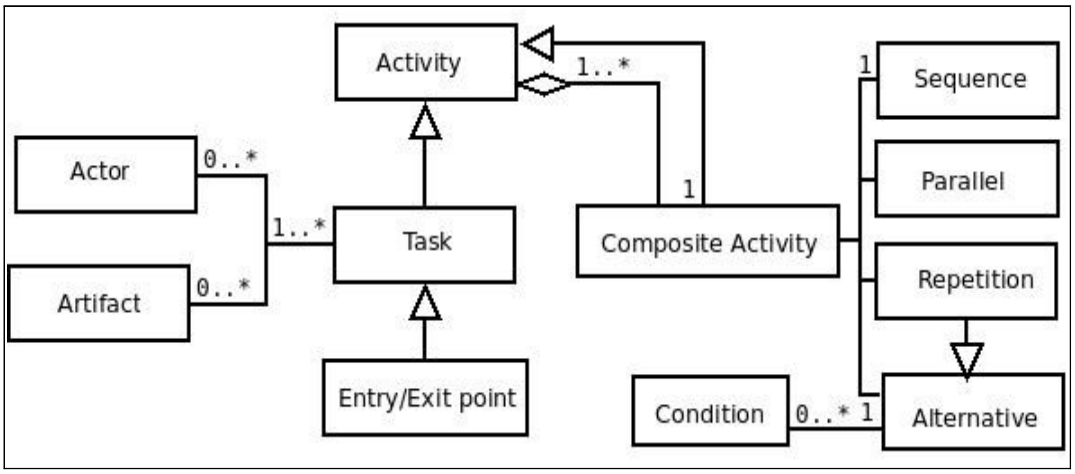


Figure 3.1.1. Classes of Process Elements.

The analysis is structured according to the types of process model elements shown in Figure 3.1.1. All cases are grouped and supplied with corresponding linguistic patterns and examples, based on scenarios from [4], [14][2] .

Linguistic patterns are to be read as follows:

- <required element>;
- [optional element];
- synonym, synonym – the same syntax, the same semantics;
- antonym (antonym) - the same syntax, different semantics;
- Activity Actor;

All the parse trees for the examples were produced with Stanford PCFG parser [10]. The parse trees use Penn Treebank II tags [2] in order to mark clauses, phrases and parts of speech (POS-tags). A list of abbreviations used for the POS-tags can be found in Appendix 1.

The following analysis is not meant to be thorough and represent all the possible variations. Instead,the goal of the analysis is to put into evidence the most common patters for this kind of documents, based on the case studies that we have analyzed.

3.2 Basic elements.

3.2.1 Activities.

Activities are the main "building blocks" of a process. Activity is a verb phrase (VP), which normally corresponds to the predicate in a sentence. Activities contain **dynamic verbs**, which

2 Including two unpublished case studies: "Improving the 'IT Incident Management' process at FictOrg" and "Improving the 'Compulsory Third Party Insurance Claims' Process at New Generation Finance (NGF)", written by Jan Recker, Queensland University of Technology.

denote an action. **Composite Activity (CA)** is an Activity consisting of other activities (e.g. subprocess). **Process** is also a composite activity, situated on the highest level of abstraction. **Task** is an atomic activity, which can-not be subdivided into activities.

Pattern1. Task.

<Task>

Call the help desk. (1P[3])

Pattern2. Composite Activity.

<Activity> [CONJ[4]] <Activity>

Composite Activity consists of several Activities, which can be connected with a conjunction.

Call the help desk [and] make a request. (2P)

Call and make.

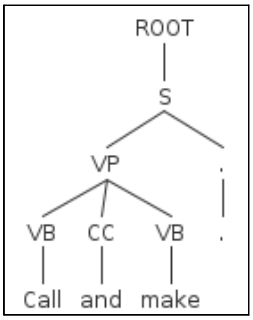


Figure 3.2.1. Simple Tasks (CA).

Call the help desk and make a request.

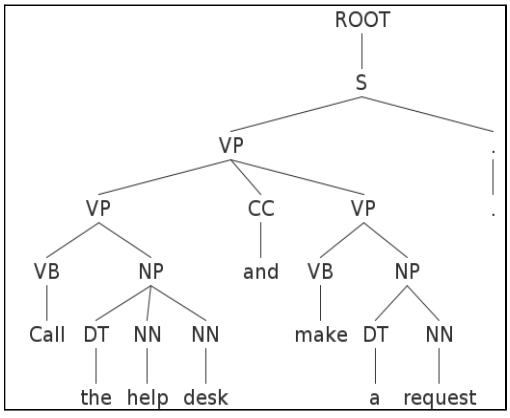


Figure 3.2.2. Complex Tasks (CA).

3 P – predicate.

4 Conjunction (e.g. and, or, for, if, in case, after, because, while, etc.)

3.2.2 Artifacts.

An Artifact is a part of an Activity and plays a role of an ***object*** in a sentence. An Artifact can denote an object, place, reference to another Actor, etc. We can generally recognize an Artifact by observing that it is a noun phrase (***NP***) which is part of **VP** of a corresponding Activity. Below we present a collection of patterns for identifying Artifacts. Each pattern is accompanied by an example.

Pattern 1. Simple Artifact

<Activity Artifact>

Corresponds to simple object, consists of one independent noun or pronoun.

Call **the help desk**. (1P)

Pattern 2. Complex Artifact.

<Activity Artifact [and, or] Artifact>

Complex Artifact consists of several simple Artifacts. Corresponds to compound object, consists of several independent nouns or pronouns.

Contact customer via **email** [or] **telephone**. (1P)

Pattern 3. Shared Artifact.

<Activity> [and, or] <Activity [Artifact]>

Warning! *This case is ambiguous, as it is not explicit, whether the Artifact belongs to one activity or to several of them. In this case, existence of supplementary "signal words" (like “both”, “only”, etc) helps to resolve the ambiguity.*

[Both] ask [and] [only] answer questions via **email.**

Both ask and answer questions via email. Ask and only answer questions via email.

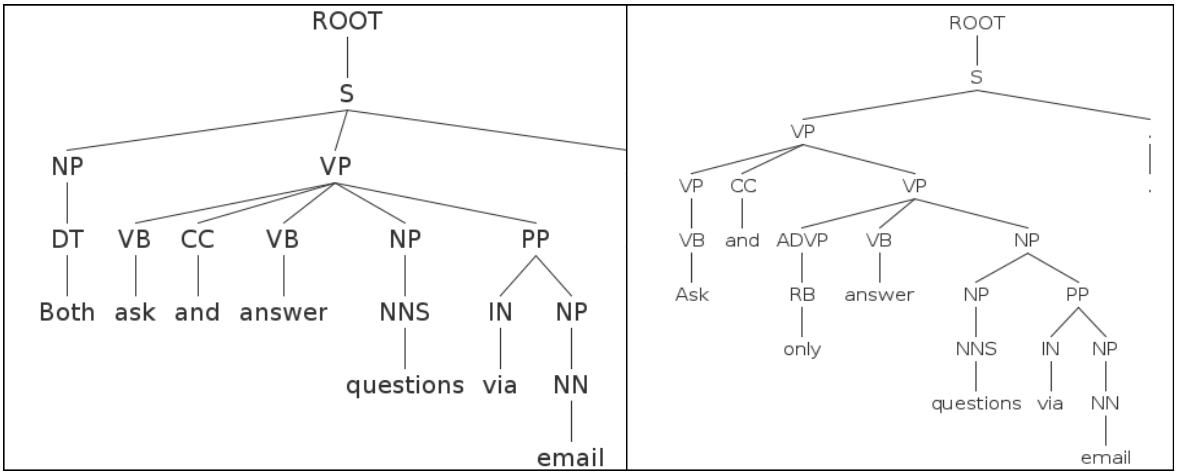


Figure 3.2.3. Two Tasks and one Artifact.

Pattern 4. Independent Artifacts.

<Activity [Artifact]> [and, or] <Activity [Artifact]>

Call **the help desk** [and] make **a request**.

3.2.3 Actors.

Actors are the performers of the Activities. They answer a question "Who is performing the Activity?", and normally play the role of a ***subject*** in a sentence and correspond to noun phrases (***NP***).

Pattern 1. Single Actor.

[Actor] <Activity>

A client calls the help desk. (1S-1P[5])

A client calls the help desk and makes a request. (1S-2P)

5 S – subject; P – predicate.

A client calls the help desk and makes a request.

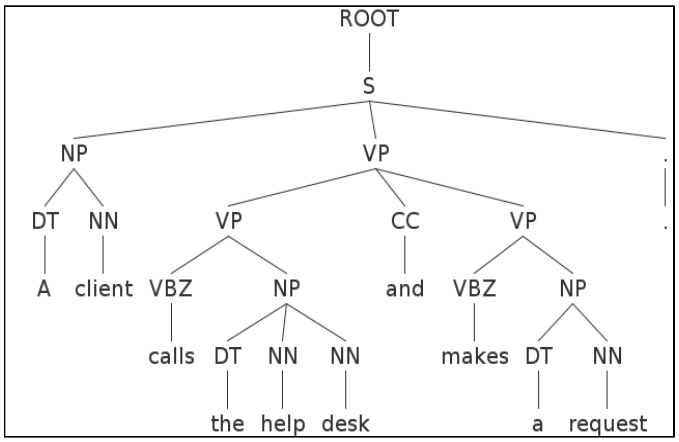


Fig.3.2.4. One Actor and two Tasks.

Pattern 2. Group of Actors.

<Actor> [and, or] <Actor> <Activity>

Several Actors performing one Activity. Actors can do the same Activity together or in parallel.

[Both]**Computer Support Officer** [and] **Desktop Support Officer** solve the service failure. (2S-1P)

Computer Support Officer and Desktop Support Officer solve the service failure.

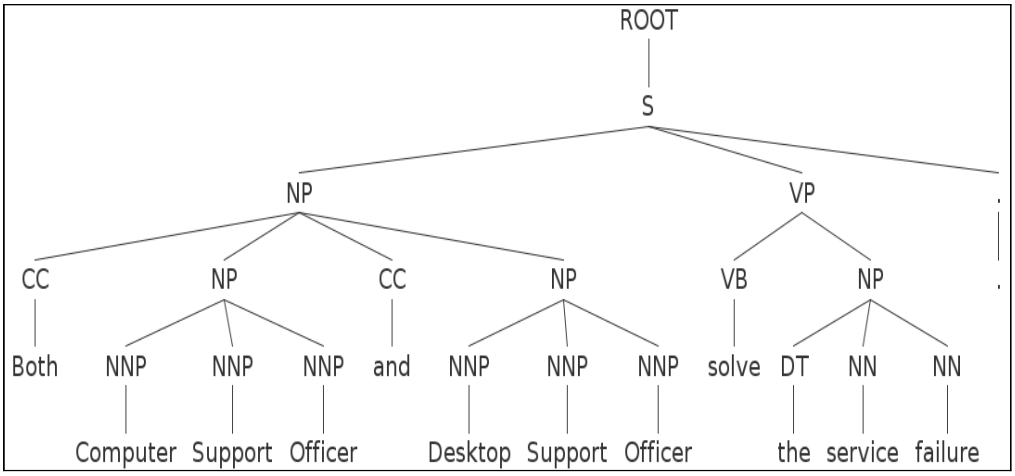


Fig.3.2.5.Two Actors and a single Task.

Pattern 3. Independent Actors.

<Actor> <Activity> [and, or] <Actor> <Activity>

Several Actors performing different Activities. Relations between Activities are regulated by a coordinating conjunction and other "signal words"(see Control Flows). ***Warning!*** *It is often crucial to have "," before "and" in this construction to be correctly parsed by the parser.*

Receipt department returns the good to the vendor [and] **system** sends notification to the purchase department. (2S-2P)

Receipt department returns product, and system sends notification to purchase department.

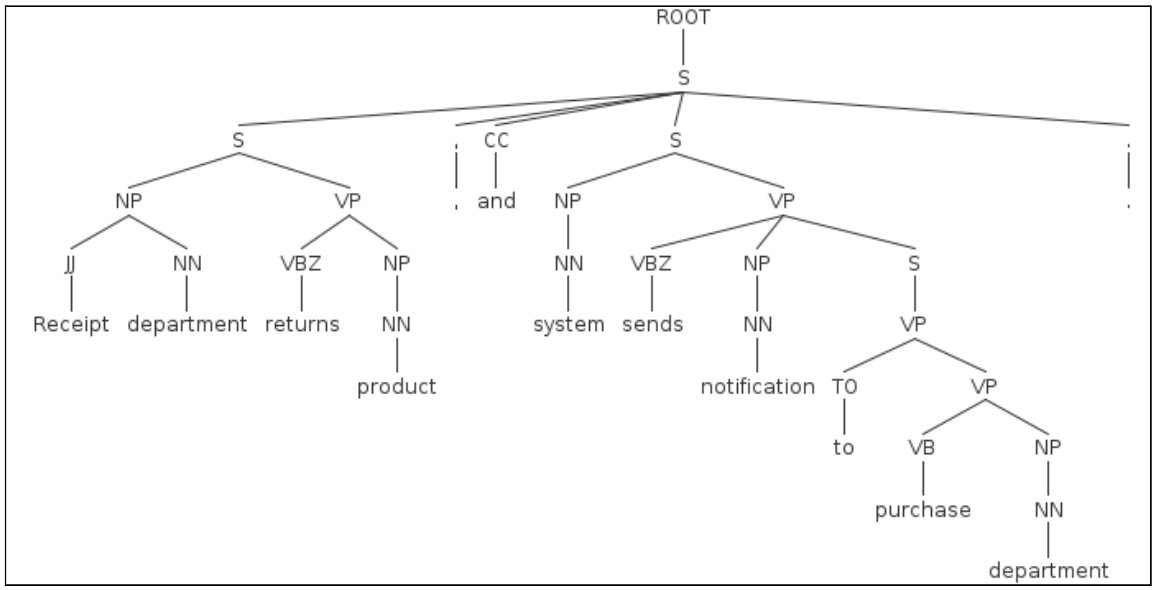


Figure 3.2.6. Two Actors and two Tasks.

Pattern 4. "Inanimate Actor".

Warning! *Sometimes inanimate object plays a role of a subject (in anti-causative structure with intransitive verbs: e.g. result, occur, etc), in this case it has to be identified as an Artifact, not an Actor of the Activity.*

An accident **occurs**.

3.2.4 Special case: Passive Voice.

Passive voice is a grammatical structure in which Artifact and Actor exchange their places: **Artifact** plays a role of a **subject** in the sentence and **Actor** – an **object**. Passive voice structure is very common for process description texts. Often when Actor is absent, it aims to focus attention on the Artifact instead.

<Artifact> <Activity [by Actor]>

Passive voice is usually formed with "be,get+past participle" construction.

The good **is returned** to the vendor [**by** receipt department**]**.

The good is returned to the vendor by receipt department.

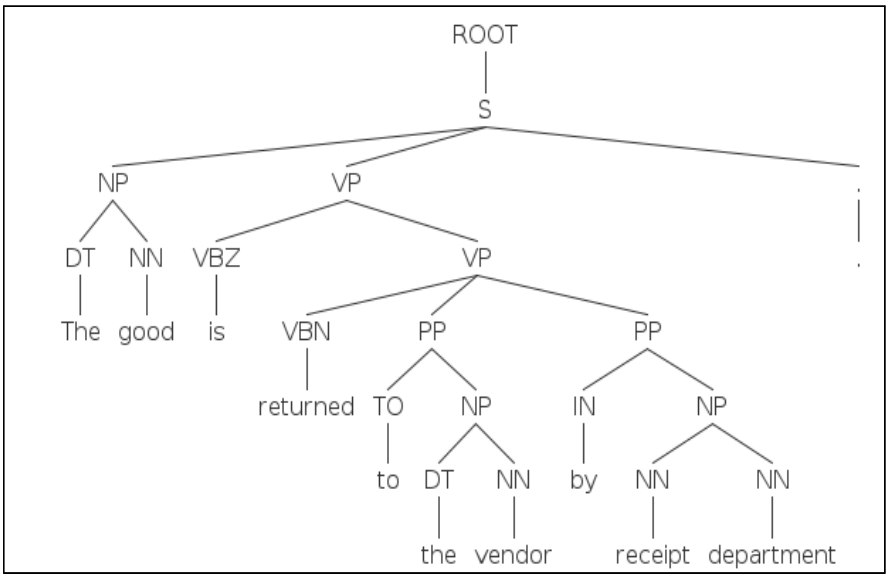


Figure 3.2.7. Passive voice construction.

Receipt department returns the good to the vendor.

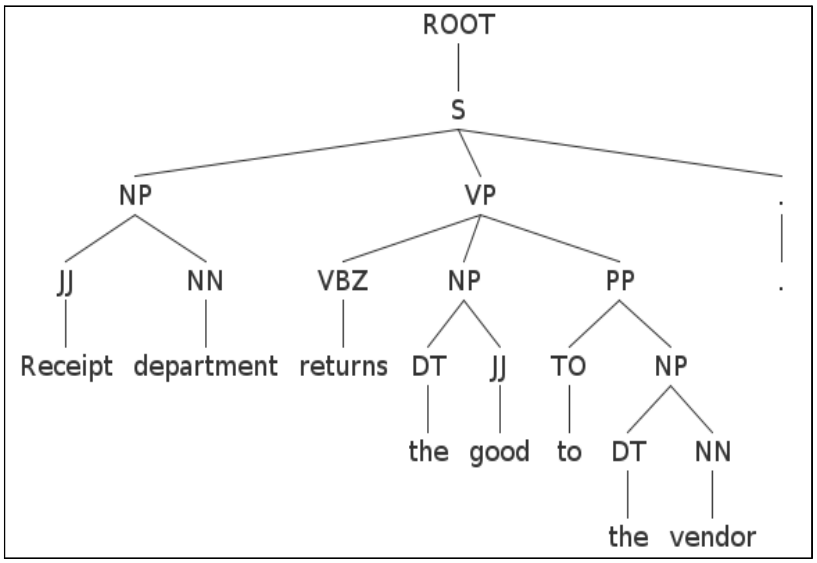


Figure 3.2.8. Corresponding active voice construction.

Warning! The word “good” in the parse tree above was incorrectly parsed as an Adjective, instead of a Noun as in the previous tree. The adjective “receipt” wasn't correctly parsed in the first tree, in contrast (looks like PCFG algorithm works more precise while parsing subjects, than objects). It is just a single illustration of possible parsing errors.

3.3 Entry and Exit Points.

Entry and Exit points mark the beginning and the end of a Composite Activity. The common "signal words" for Entry(Exit) point are respectively: **start**, begin, trigger (**end**, finish, complete, terminate).

Pattern 1. Simple entry/exit.

<Composite Activity> **start**, begin(**end**, finish, complete, terminate) [with(when), <Activity>]

Entry/Exit point description contain reference to the Composite Activity, which they belong to. Additionally the first Activity (or reference to it if it was previously described) can be specified. Before this Activity entry to the CA is made.

When relevant forms are sent, the Notification process **ends** without any further handling specified.

Request is marked as complete and the process **ends**.

The process starts.

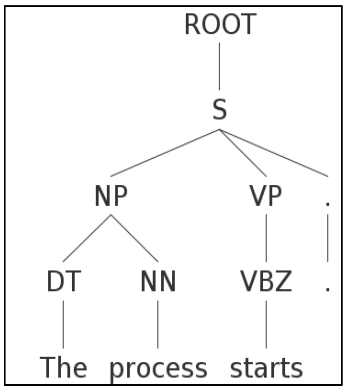


Figure 3.3.2. Entry point to CA.

The incident management process **starts with** the detection of a service failure.

This task **terminates with** the actual delivery of the freight to the customer.

This task terminates with delivery of freight.

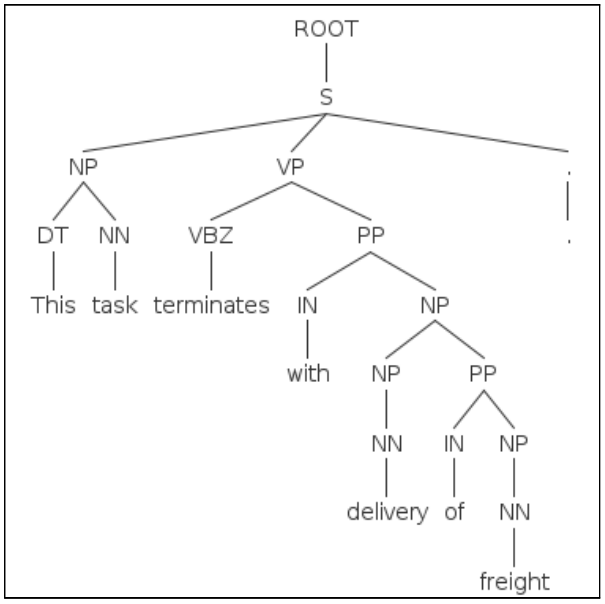


Figure 3.3.3. Exit point from CA **through** an Activity.

Pattern 2. "Signal Activity".

<Activity> **start**, begin, trigger (**end**, finish, complete, terminate) <new(current) CA>

The word "trigger"("change to") indicates both Exit point from previous Composite Activity and entry point to the new Composite Activity.

The reception of relevant claim documentation **triggers** The Claims Creation process.

The detection of a service failure **ends** the Claims Management process.

Detection of service failure ends Claims Management process.

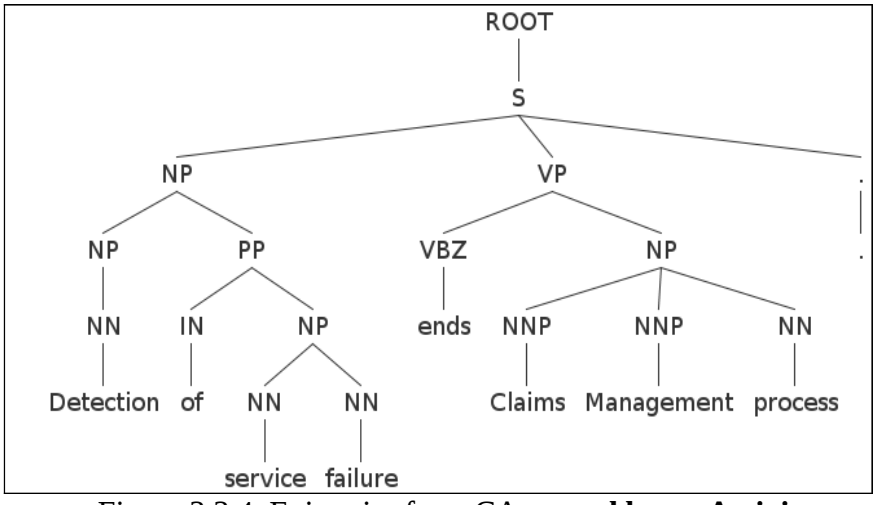


Figure 3.3.4. Exit point from CA **caused by an Activity**.

Pattern 3. Actor initiating Entry/Exit.

<Actor> **start**, begin (**end**, finish, complete, terminate) <Activity>

The Service Desk **starts** the Incident resolution process.

Service Desk starts Incident resolution process.

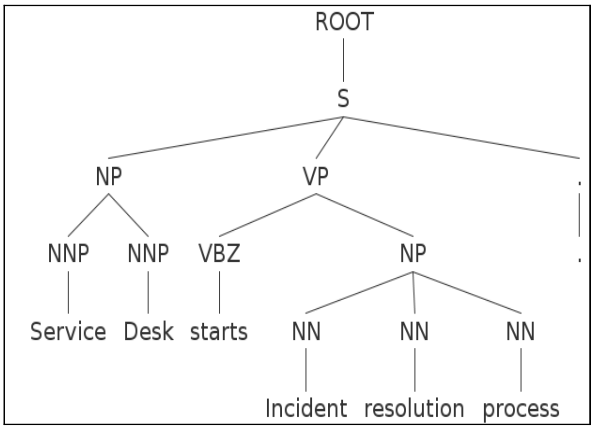


Figure 3.3.5. Entry point **caused by an Actor**.

3.4 Control Flows.

Control flows regulate relationships between activities and changes of a workflow in a Process. There are 4 types of relationship between activities (control flows):

- Sequence
- Concurrency
- Choice
- Repetition

They are similar to the sentence semantic patterns proposed in [9].

3.4.1 Sequence.

Sequence control flow indicates that Activities are happening “one after another”, in sequential order.

Pattern 1. Direct order.

<previous Activity> [then, after, afterward(s), later, next, and(**!**)] <new Activity>

This signal words additionally stress the sequential order. They often can be omitted or replaced with comma (“,”).

Warning! *“And” can also indicate Concurrent flows. Other "signal words" for*

Concurrent flows(see Concurrency) accompanying "and" can help to avoid ambiguity in this case.

A client calls the help desk [, **and**] makes a request. (1S-2P)

Receipt department returns the good to the vendor. **After**, system sends notification to the purchase department. (2S-2P)

Pattern 2. 1. Custom order (after).

after, upon, once, when <previous Activity>, <new Activity>

Clauses can be rotated, "after" always indicates previous Activity.

After an accident occurs, the Administration Team receives advice of the accident via telephone.

When a PO Manager creates a PO, the system initiates the task automatically.

The file is referred to Legal Support, **once** judgement is received.

When Manager creates order System initiates task.

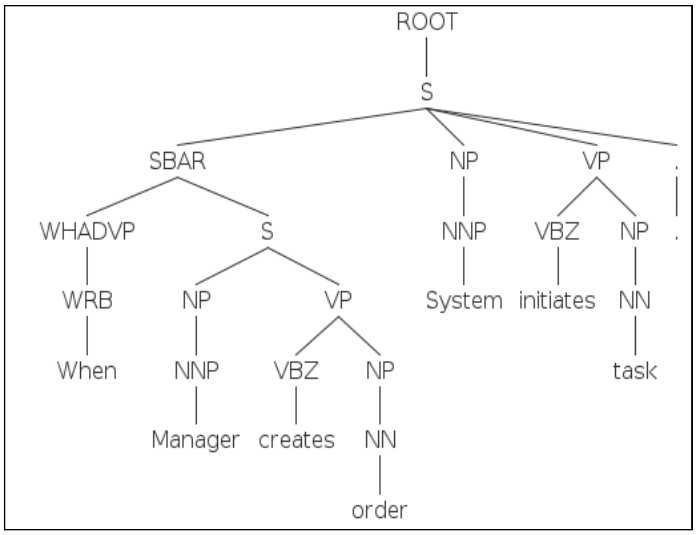


Figure 3.4.1. Time clause indicating **Sequence** flow.

Pattern 2. 2. Custom order (before).

before <new Activity>, <previous Activity>

Clauses can be rotated, "before" always indicates new Activity.

Before Supply Officer creates a Shipment Information document, the appointment is made.

The appointment is made **before** Supply Officer creates a Shipment Information document.

Pattern 3. Reference to Activity.

> ***Warning!*** *Activity can be a reference to the other Activity, which was already described before, and do not present a new Activity (e.g.* lead to, result in, etc.*)*

PO is confirmed and the confirmation letter is sent to the customer. **After** confirmation of a PO, a route guide needs to be prepared and the trailer usage needs to be estimated.

Officer notifies Problem Manager to confirm the severity. This **results in** a confirmation of Severity 1.

After **confirmation of PO**, prepare route guide.

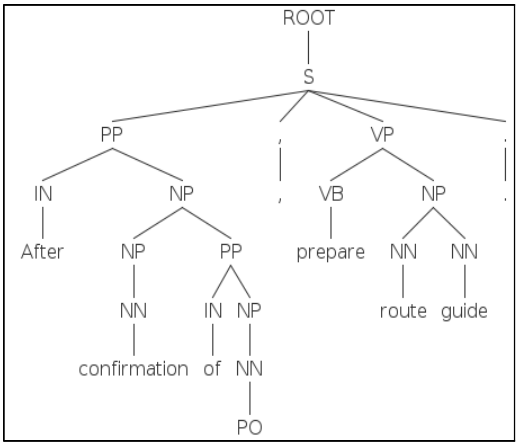


Figure 3.4.2. Reference to previous Activity **by a Noun Phrase**.

This results in confirmation of Severity 1.

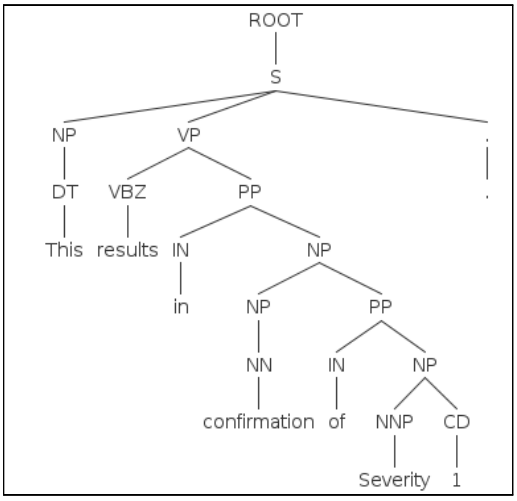


Figure 3.4.3. Reference to previous Activity **by a demonstrative**.

Pattern 4. Explicit numbering.

Often explicit numbering of Activities(first, second, third; A, B, C) and text structure (e.g. numbered or bulleted list) are used to stress the sequential order between activities.

3.4.2 Concurrency

Concurrent control flow indicates that 2 or more activities happen in parallel, in the same time. Concurrent flow is modeled by parallel AND-split.

Pattern 1. While.

while <Activity>, <Activity>

While Officer log entries, Client Liaison address customer inquiries.

While Officer log entries, Client Liaison address customer inquiries.

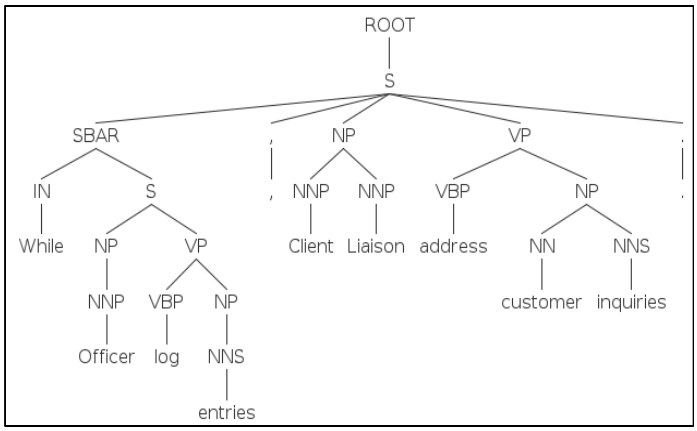


Figure 3.4.4. Time clause indicating Concurrent flows.

Pattern 2. AND.

<parallel Activity> [and] [independently, simultaneously, at the same time, in parallel] <parallel Activity>

Warning! “And” can also indicate *Sequence* flow. *Additional* "signal words" help *to explicitly indicate parallelism between activities*.

Officer checks the invoice [**and**] [**simultaneously**] fills the form. (1S-2P)

ENIC verifies the degrees [**and**] [**in parallel**] an officer checks student's test on-line. (2S-2P)

3.4.3 Alternative.

Alternative is a type of control flow, which divides the workflow into several alternative workflows.

Pattern 1. Ambiguous OR.

<alternative Activity|Actor|Artifact> or <alternative Activity|Actor|Artifact>

Coordinating conjunction (CC) “or” presents alternative elements (activities, actors or artifacts).

Warning! Coordinating conjunction “or” doesn't explicitly indicate which of 2: OR-split or XOR-split is meant(ambiguity). It is very difficult to resolve this ambiguity not knowing the meaning(semantics) of the words. Additional "signal words" can help to automatically identify type of the split.

Warning! *CC “or” can also identify another name of the same element, not a new element.*

System automatically raises a new incident record on the SOLVE system, **or** notifies the Resolver Group of the service failure. 1S-2P

CSO **or** a Workflow Controller handle assigned incidents. 2S-1P

Contact customer via email **or** telephone.

System **raises incident record** or **notifies Resolver Group**.

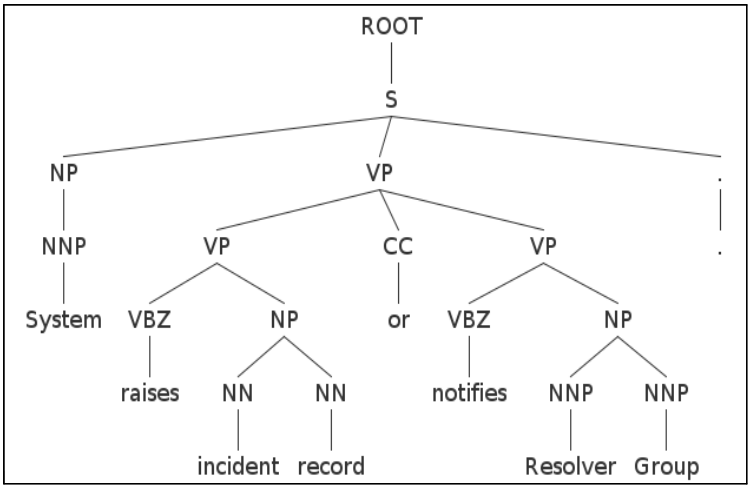


Figure 3.4.5. Alternative **Activities**.

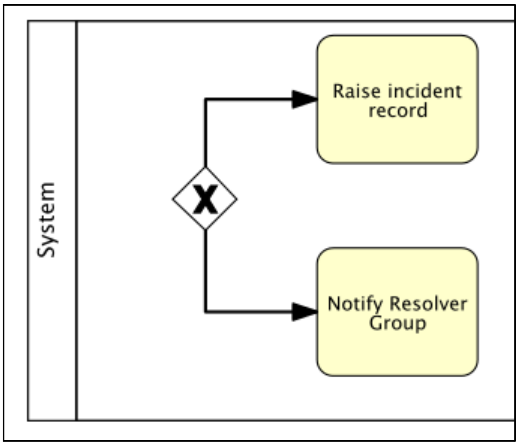


Figure 3.4.6. Alternative **Activities** in BPMN[6].

6 All BPMN diagrams included in Chapters 3, 4 were produced with ORYX-editor [22].

CSO or **Workflow Controller** handle incidents.

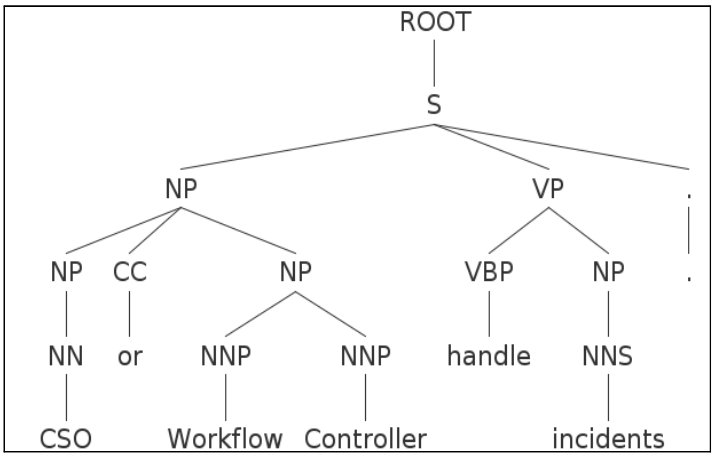


Figure 3.4.7.Alternative **Actors**.

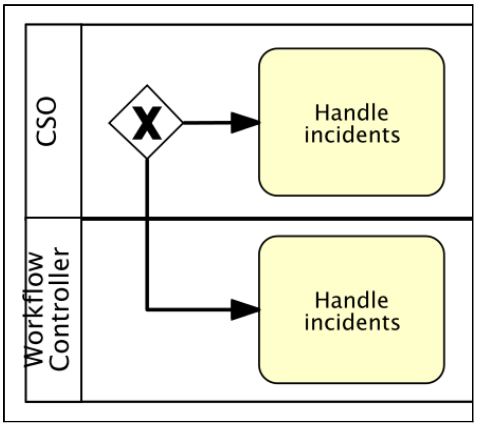


Figure 3.4.8. Alternative **Actors** in BPMN.

Contact customer via **email** or **telephone**.

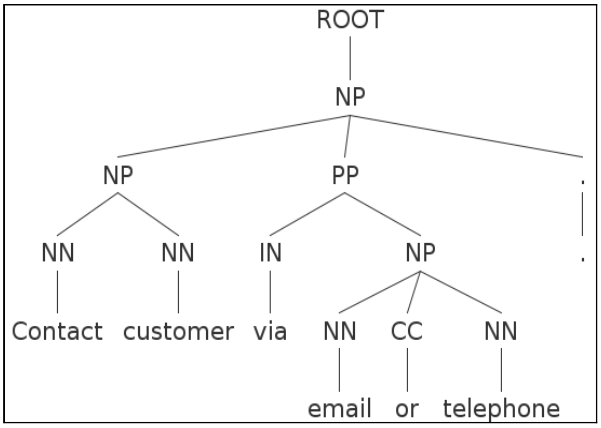


Figure 3.4.9. Alternative **Artifacts**

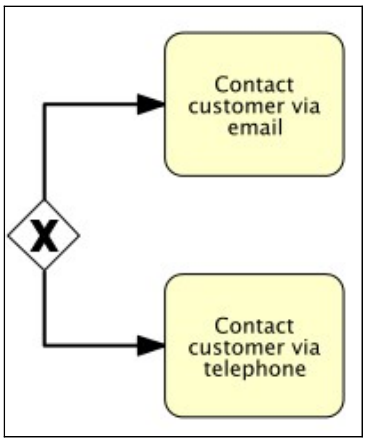


Figure 3.4.10. Alternative **Artifacts** in BPMN.

Pattern 2. Exclusive OR (**XOR**)**.**

either, only <alternative1> or <alternative2, (**no(t)** alternative1)>

Exclusive XOR-split can be explicitly identified with the help of "signal words", which mark the first exclusive alternative, "or" identifies the following exclusive alternative.

*Alternatives can be **explicitly exclusive** using negation ("confirm or do not confirm") or **implicitly exclusive** using antonyms ("confirm or reject"). In the later case, to identify such alternatives a list of the corresponding antonyms must be handled.*

Officer **either** confirms PO or rejects.

Officer either confirms PO or rejects.

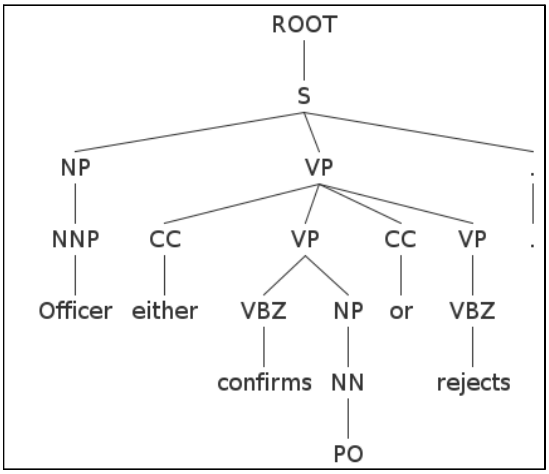


Figure 3.4.11. "Either" explicitly indicates XOR-split.

Pattern 3. Inclusive OR.

<alternative1> and/or <alternative2>

Inclusive OR-split is a merge of XOR- and AND- splits, which indicates possibility of both Alternative and Concurrent flows and can be explicitly identified with the help of the compound conjunction "and/or" ("or/and").

A client gives a feedback on the previous order **and/or** makes a new order.

Pattern 4. IF.

if, whether, in case [of], till, until (unless) <Condition>, <alternative Activity>

*Conditional clauses is a common method to introduce alternative Activity(s), which indicates a **split** into 2 or more separate flows immediately after it.*

There are cases with only one alternative Activity, which is also called optional Activity. In further descriptions we will use the term "alternative" for both optional and alternative Activities, meaning that optional Activity has an alternative Activity, which is empty.

Position of If-clause can rotate, but most often it is in the beginning of a sentence, preceding the description of a corresponding Activity(s).

In case the system detects a service failure, it automatically raises a new incident record on the SOLVE system.

If PO is confirmed, a route guide needs to be prepared and the trailer usage needs to be estimated.

Unless PO is confirmed, [do not] prepare guide.

If PO is confirmed prepare route guide and evaluate trailer usage.

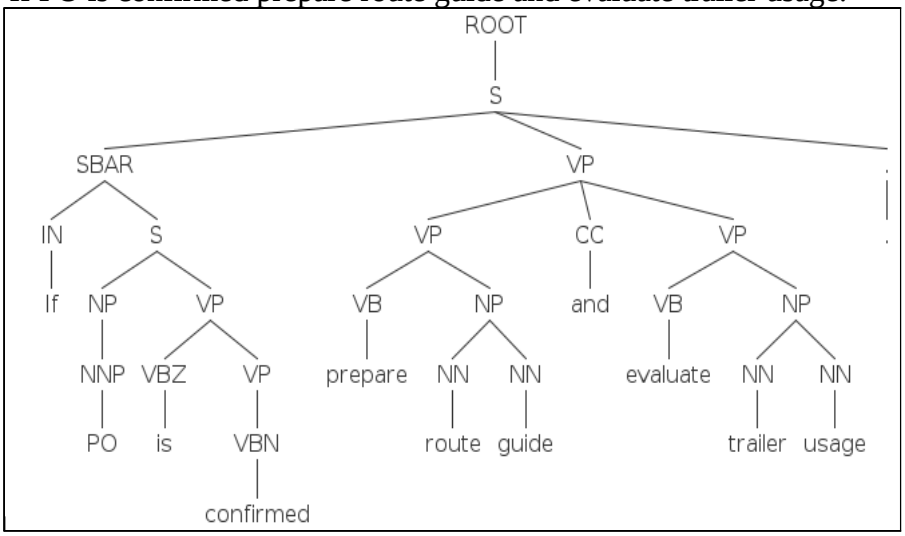


Figure 3.4.12. Conditional clause indicating **Alternative** flows.

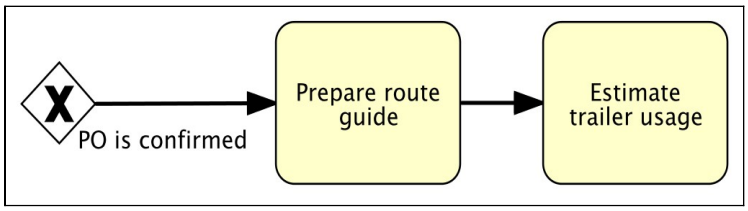


Figure 3.4.13. Fragment of **Alternative** flow created **by If-clause**.

Pattern 5. IF as a reference.

if, whether, in case <previous Activity>, in [latter[7]] case [of], <new Activity>

Warning! *Conditional clause(e.g. If-clause) can contain a* ***reference to a previous Activity****, described before, then it identifies Sequence not Alternative (see Sequence).*

PO is confirmed or rejected. **If** PO is confirmed, a route guide needs to be prepared and the trailer usage needs to be estimated.

In case of a temporary work-around, a SOLVE Service Request for a long term solution is raised based on the SOLVE Problem Record.

In the latter case, the Service Request Process is triggered which is not in scope of Incident Management. **In the former case**, the Resolve Group checks if the incident is correctly assigned to an appropriate group.

In case of work-around, raise request.

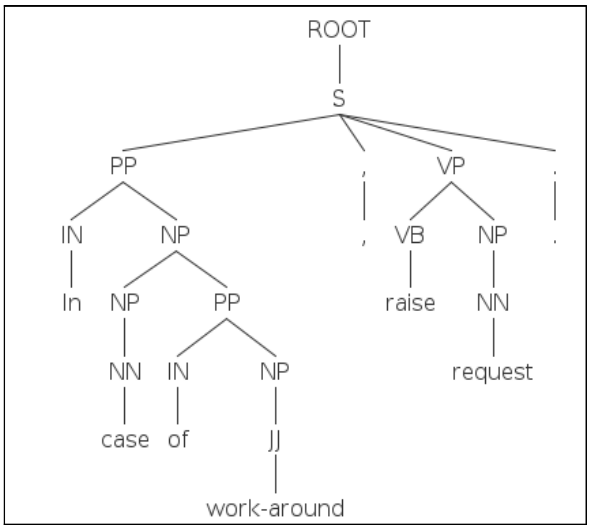


Figure 3.4.14. “In case of” construction.

7 this/that, first/second/..,former/later.

PO is confirmed or rejected. If PO is confirmed, prepare route guide and estimate trailer usage.

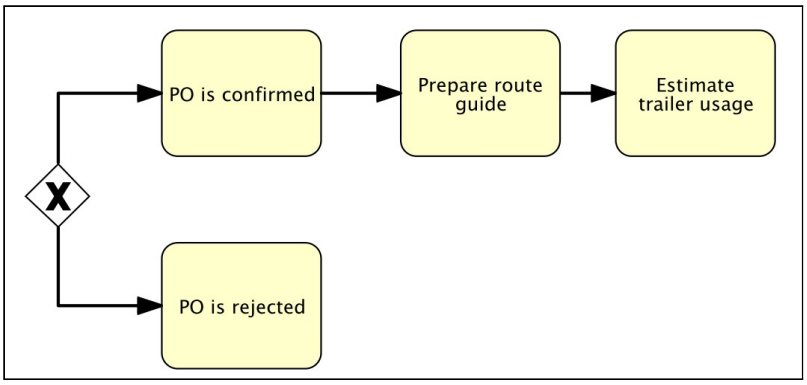


Figure 3.4.15. **Reference to the previous alternative Activity** by If-clause.

See the difference with the Figure 3.4.13.

Pattern 6. "If..if"-join.

if, whether, in case [of], till, until (unless) <Activity1>
[and, or]
[if, whether, in case [of], till, until (unless)] <Activity2>, [then] <Activity3>

*Coordinating conjunction between several conditional clauses indicates one of the **joins**:*

- *"and" - **AND-join**;*
- *"or" - **OR-join** or **XOR-join**(ambiguity – see "Or" section above);*

If an incident has been handed-off to a Resolver Group **and if** an incident has been handled as outlined above, a SOLVE Problem Record is created.

If the resolution was rejected **or** updates were made, the resolution is reconsidered.

If incident was handed-off, and if incident was handled, create record.

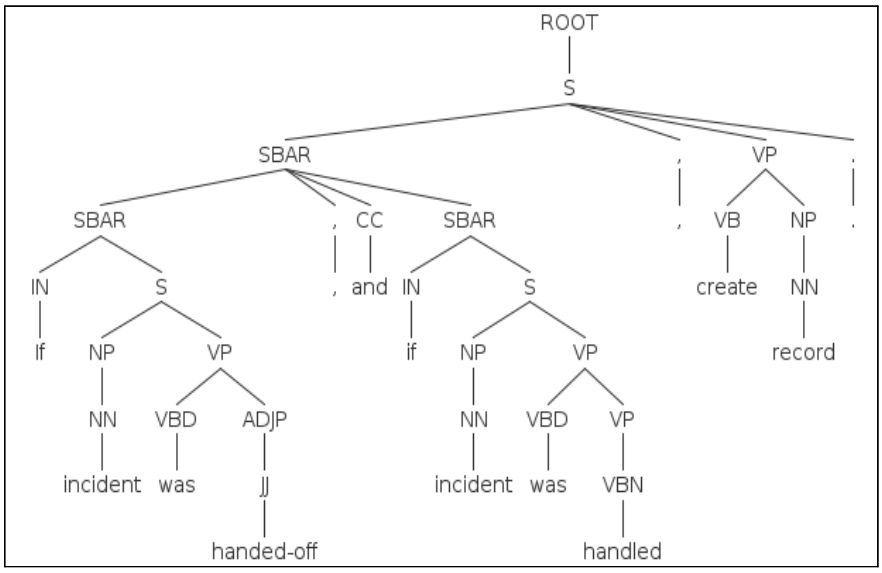


Figure 3.4.16. AND-join indicated by **two If-clauses**.

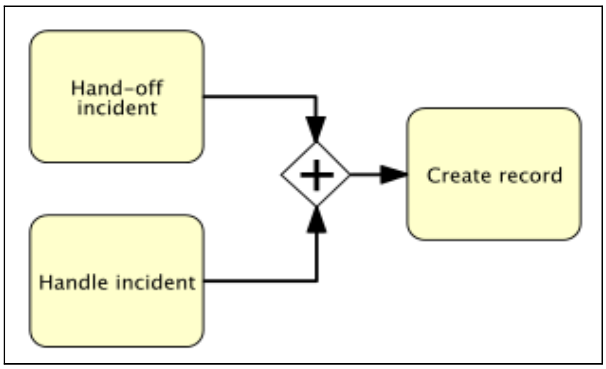


Figure 3.4.17. **AND-join** indicated by two If-clauses in BPMN.

If the resolution was rejected or updates were made, the resolution is reconsidered.

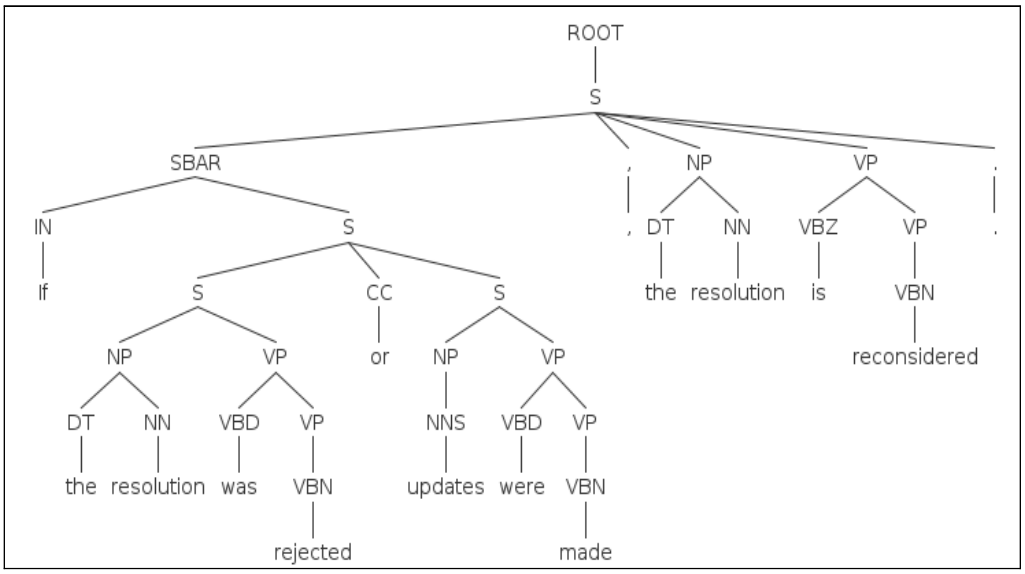


Figure 3.4.18. OR-join by **one complex If-clause**.

Pattern 7. "In..case"-join.

[in] either, both, all case[s] <Activity>

Construction "in..case(s)" indicates OR-join.

In <u>all other cases</u>, an incident record is raised in the SOLVE – Problem system.

If resolution doesn't satisfy, <u>reject resolution</u> or <u>make updates</u>. **In <u>both cases</u>** resolution is reconsidered.

In all other cases, raise incident record.

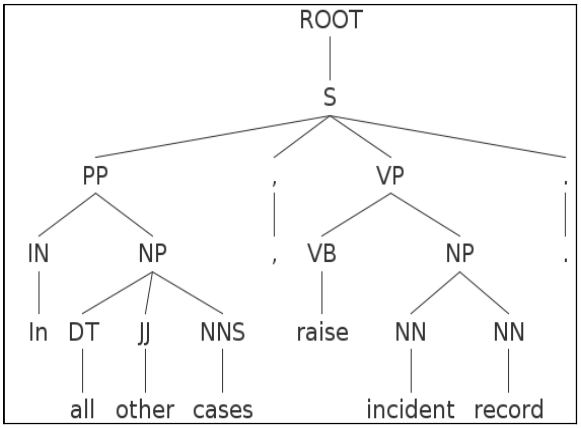


Figure 3.4.19. "In..case(s)" construction.

If resolution doesn't satisfy, reject resolution or make updates.

If resolution was rejected or updates were made(in both cases), resolution is reconsidered.

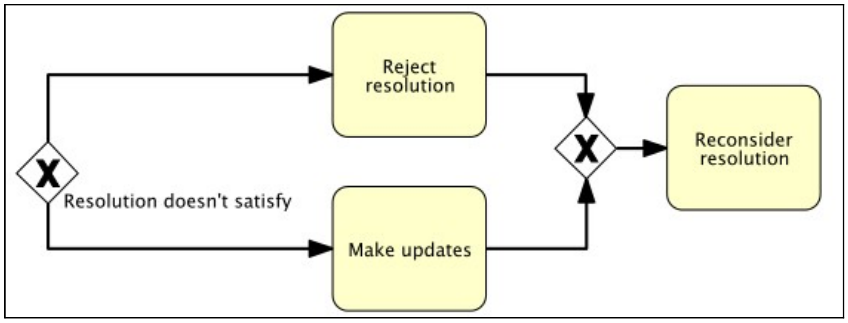


Figure 3.4.20. **OR-split** and **OR-join**.

Pattern 8. “False IF”.

> ***Warning!*** *When conditional clause (e.g. If-clause) is a part of an Activity (object clause), then it doesn't indicate Alternative activity.*

The Resolve Group checks [**if** the incident is correctly assigned to an appropriate group].

Decide **whether** a SOLVE Problem Record is created.

After communicating the resolution, evaluate [**if** the resolution was permanent **or if** additional action is required].
If the service was not restored determine [**if** the incident requires re-assignment].

If the service was not restored determine if the incident requires re-assignment.

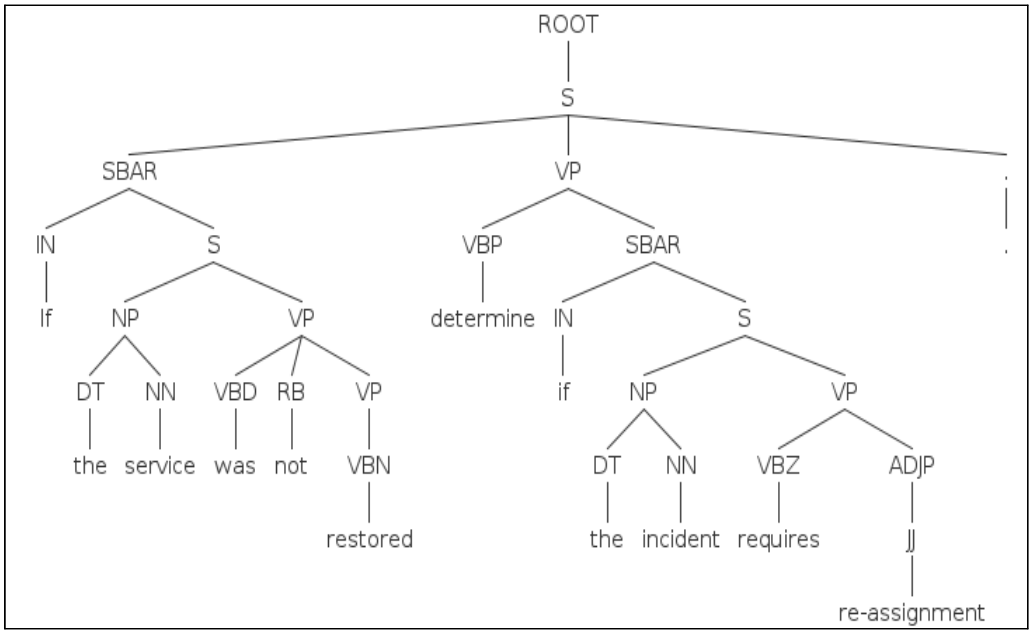


Figure 3.4.21. If-clause as an object clause.

Pattern 9. Else.

else, otherwise, if not, <alternative Activity>

If it is urgent, the employee would "hand deliver" the form, **otherwise** it would go via internal mail.

Otherwise send via mail.

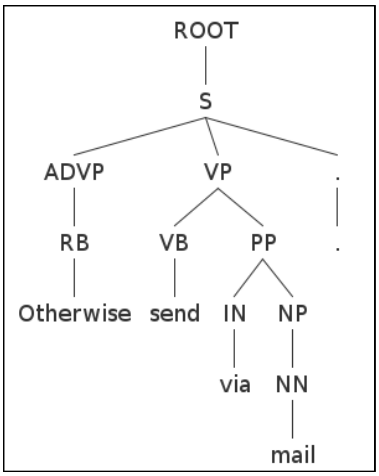


Figure 3.4.22. **Alternative** activity indicated **by adverb**.

Warning! Constructions **"**If.., otherwise..", "else", "if not" often are not correctly parsed by the parser.

Pattern 10. Option (alternative).

<Actor> have (provide, etc) option (alternative) <alternative Activity>

Client have an **option** to notify the incident management team of the failure.

FictOrg provides an **option** to access SOLVE system.

FictOrg have an option to notify SOLVE system.

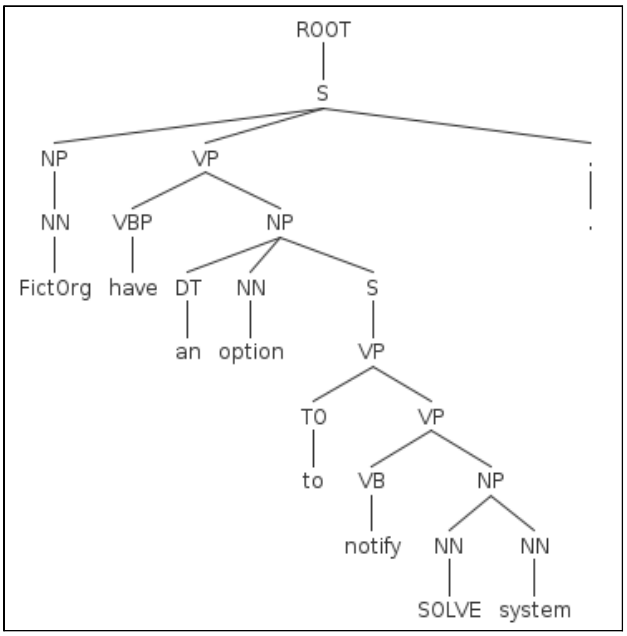


Figure 3.4.23. Explicit optionality.

Pattern 9. Can, may.

<Actor> can, may <Task>

Client **may** detect a service failure.

Client may detect a service failure.

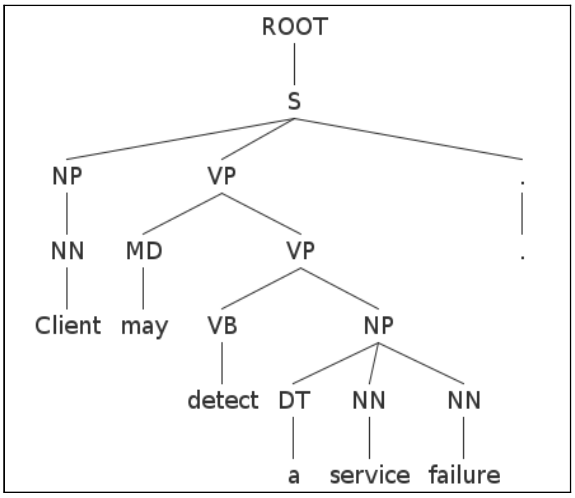


Figure 3.4.24. “Modal” optionality.

Pattern 9. Appropriate.

if [it is] appropriate, applicable, optionally, not necessarily, <alternative Action>

If applicable, an RVA needs to be recorded in the Meridian Mai IVR system.

Optionally Service Desk team raise a new incident record.

Optionally Service team raise new incident record.

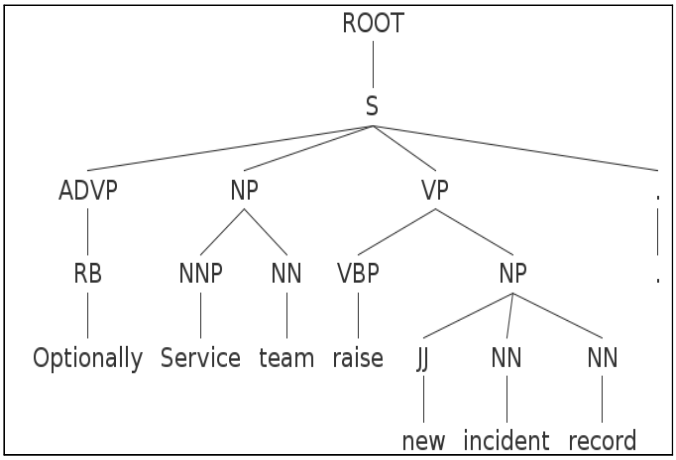


Figure 3.4.25. Explicit optionality by adverb.

3.4.4 Repetition.

Repetition is a special case of Alternative, when one of the alternatives is the same Activity, that was before the Condition. Hence activity makes a loop on itself through a Condition. If condition is empty, the loop becomes infinite. In order to avoid an infinite loop in the process, number of repetitions has to be indicated explicitly or via a condition for repetition to stop, otherwise we assume that Activity has to be repeated only once.

Warning! Repetition is often assigned through **references** to previous Activities (the same function as of "go-to" statements in programming languages).

Pattern 1. Ambiguous Repetition.

> *If Repetition is not accompanied with "signal-words", which indicate a Condition for Repetition to stop (Exit point from Composite Activity with Repetition control flow), it is ambiguous when Repetition has to stop. In order to avoid an infinite loop in the process, it is safe to assume, that in this case repetition has to be made only once.*

Pattern 1.1. Repeat, continue.

repeat, continue <Activity>

Repeat the assignment.

Pattern 1.2. Again.

<Activity> again

The incident is assigned and diagnosed **again.**

Go through the process **again.**

Pattern 1.3. Re-work.

re[-]<verb>

Prefix "re-" is a part of a word, which indicates a repeated Activity. In order to avoid an infinite loop in the process, Activity is repeated only once, unless there are additional "signal-words" specifying explicit number of repetition or a condition for repetition to stop.

Re-assign the incident.

Pattern 2. Controlled repetition.

Ambiguous repetition is avoided when it is supported with addition "signal words".

Pattern 2.1. Condition-controlled repetition.

if, whether, in case [of], till, until (unless) <Condition> <ambiguous Repetition>

The "signal words" for a Condition, which controls Repetition, are the same as for Alternative Flow, that were already described in the previous section.

If PO was not approved, repeat modifications.

If request is not correct, it goes through the Process again.

If the incident is not correctly assigned, re-assign the incident.

If incident is not correctly assigned, re-assign the incident.

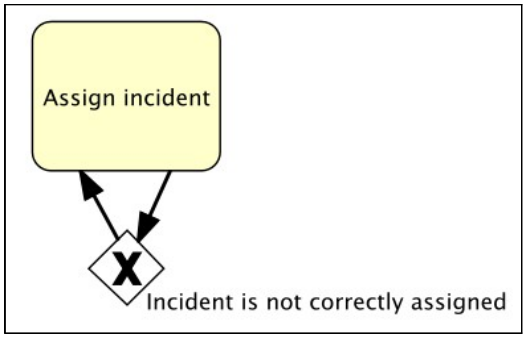


Figure 3.4.26. Repetition of Activity.

If request is not correct, it goes through the **Process** again.

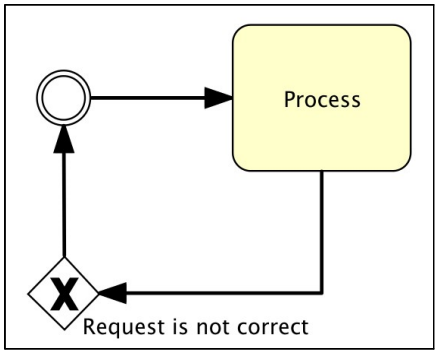


Figure 3.4.27. Repetition of **CA**.

Pattern 2.2. Count-controlled repetition.

<Activity> once, twice, <CD[8]> times

Number of repetition is explicitly assigned via multiplicative numerals(once, twice, thrice) or cardinal numbers in conjunction with a "signal word" (e.g. three times). Hence, the Condition in this case will play a role of a counter.

Print the application form **twice.**

Pattern 2.3. Collection-controlled repetition.

[for] each, all [of] <Artifact> <Activity>

Repeated action is often applied in order to process a certain number of Artifacts of the same type (a collection). Condition for an exit from Repetition in this case can be phrased as the following: "Until all <Artifacts> are <Activity>ed".

Each of the documents has to be reviewed. → Activity: "Review document";

→ Condition: "Until all documents are reviewed".

3.5 "Noise"

BP descriptions are often too lengthy and contain a lot of supplementary details, which can be irrelevant or not crucial for the understanding of a process, but produce "noise" by being false identified as process elements or making a model too lengthy and therefore less readable.

The problem is to filter out the parts of a text, which are unrelated and irrelevant for the process. However these filtered parts can be added to the corresponding process elements as supplementary comments (providing additional details), they should not interfere with the main body of the process model due to the reasons listed above.

8 Cardinal number (Penn Treebank II tag, see Appendix 1).

Pattern 1. “Noisy” sentences and clauses.

Pattern 1.1. Stative verbs.

<Actor> be, exist(have, posses, belong; contain, consist of; seem; need etc)

*A predicate can contain a **stative(state) verb,** which describes a state, not an action - Activity, but a static fact.*

The default PO Manager **is** Carmine Marino.

A PO **contains** information about the client’s company.

Pattern 1.2. “Dummy subject – wise object”.

***Warning!** Activity can be in a subordinate clause of a main clause identified as “noise”.*

It is possible that only one of these or even none of these appointments is made.

It is possible that system detects failure.

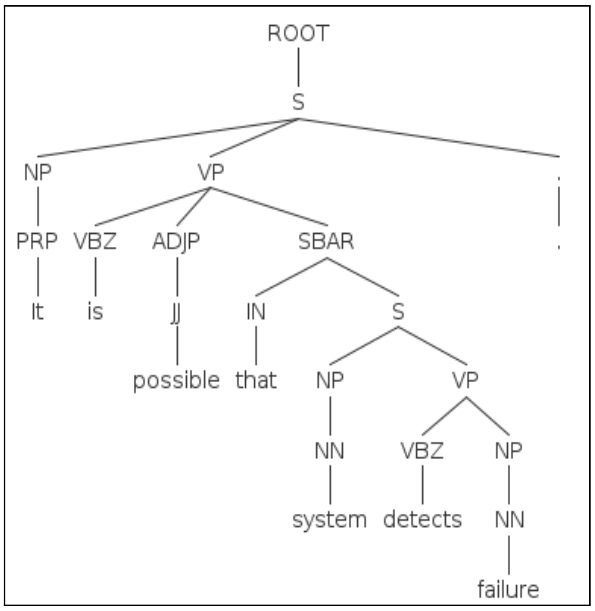


Figure 3.5.1. Activity in a subordinate clause of a “noisy” main clause.

Pattern 1.3. Subordinate clauses.

Other types of subordinate clauses, which do not contain an Activity: reason (because, since, as, given), place (where, wherever, anywhere, everywhere, etc.) clauses, clauses of manner (as, like, the way).

As they are in control of the SOLVE system they raise a new incident record.

As they are in control of System they raise new record.

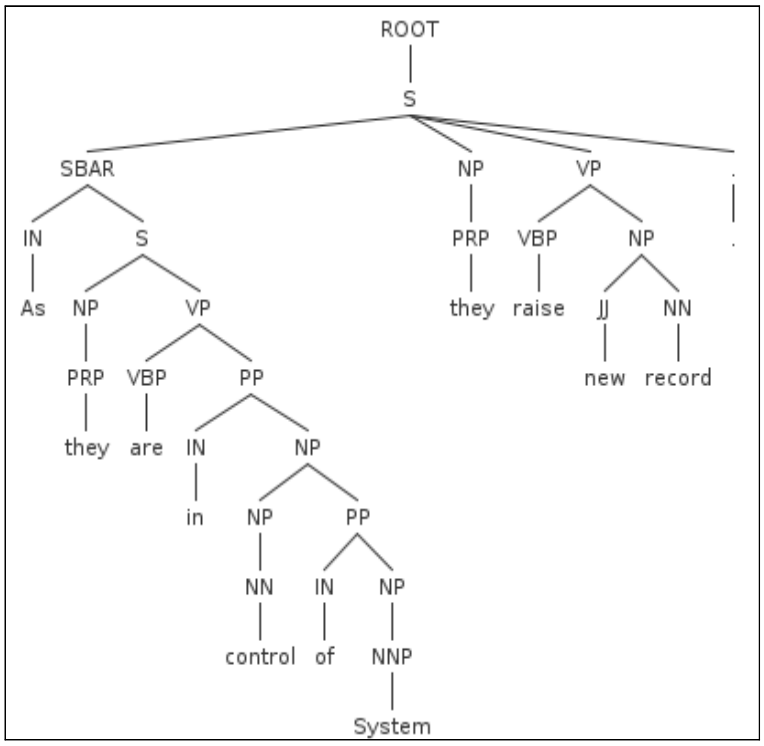


Figure 3.5.2. “Noisy” reason clause.

Pattern 2. “Noisy” words and phrases.

Pattern 2.1. Adverbs&Adjectives.

Adverb and adjective phrases (ADVP, ADJP - tags).

This results in a confirmation of Severity 1 **respectively**.

Moreover, the PO is confirmed within 3 days.

In addition, the customer notification is issued.

Also, same to final solutions, close the SOLVE record, **independent from the type of record**.

Also, same to final solutions, close the SOLVE record, independent from the type of record.

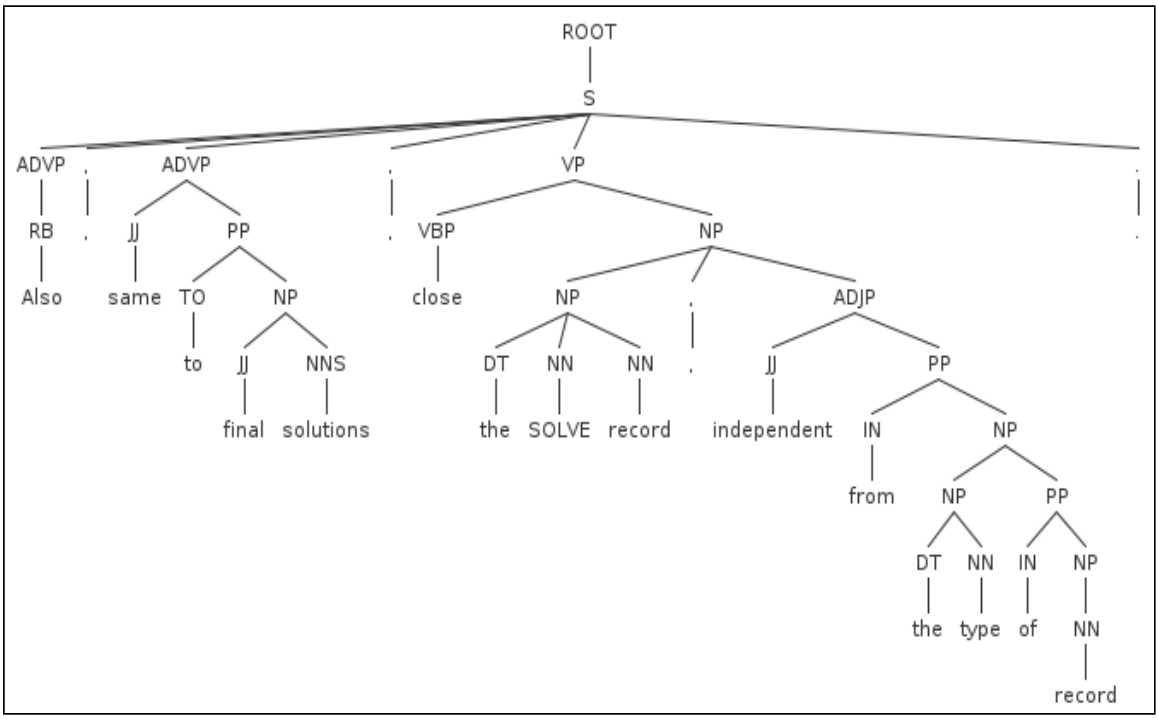


Figure 3.5.3. Adverb and adjective phrases.

Pattern 2.2. **Auxiliary verbs.**

[Actor] be (do, have, need, must, can, will, shall, etc.) <Activity>

Auxiliary verbs *are helping verbs, which do not indicate a separate activity but add some extra meaning to the next verb, which corresponds to Activity. Auxiliary verbs support formation of a passive voice structure, continuous and perfect tenses, express modality (probability, ability, permission or obligation).*

Whiteboard **needs to be** completed. → "Cleared" Activity: Complete whiteboard.

If it is urgent, the employee **would** "hand deliver" the form, otherwise it **would** go via internal mail.

Whiteboard needs to be completed.

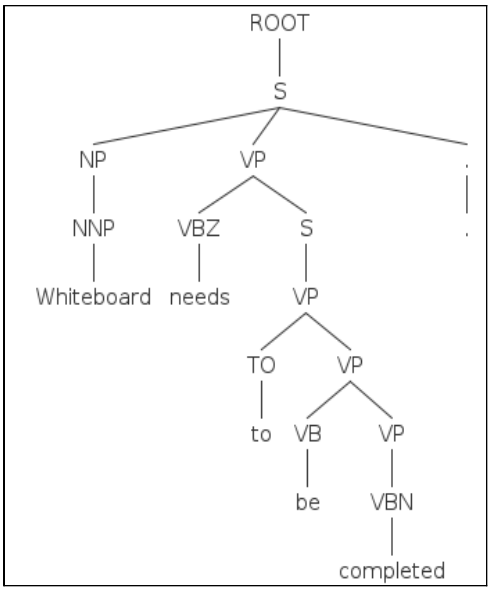


Figure 3.5.4. Auxiliary verbs.

Pattern 2.3.**Articles.**

Definite and indefinite articles (a, an, the - DT tag) do not contain important information for the process model and can be eliminated.

4 Implementation

In order to verify the approach to semi-automated extraction of process elements and relations between them from plain-text business process descriptions, a software prototype was implemented and validated. It is based on a subset of the rules, which were identified during analysis phase and previously described in Chapter 3.

Basic assumptions:

1. One input file contains only one process (the program does not currently deal with identifying several separate processes in an input text);
2. Sentences in the narrative are ordered sequentially unless the other order is explicitly indicated by certain "signal words".

4.1 List of Functions

Identify and extract Process elements:

1 Activities (Composite Activities and Tasks);

2 Actors;

3 Artifacts;

4 Entry&Exit Points("signal words": **start/begin, finish/end/terminate/complete** + **with**);

5 Control Flows:

5.1 Sequence (**default**);

5.2 Parallel ("signal words": **while**);

5.3 Alternative ("signal words": **or**, **if**, **can/may, else/otherwise, until/till**);

5.4 Repetition ("signal words": **repeat/continue, again/twice/ CD times**).

- Passive Voice support: identify and transform **passive voice** into active.

Stop words, used in order to clean the process model from “noise”:

- Articles: a, an, the;
- Auxiliary verbs: will, would, should, must.
- Parentheses with their contest

  (we assume, that parentheses contain irrelevant for the process model details);
- “Signal words” for Sequence: then, when, after, upon, once

  (they are not needed, as Sequence order is identified by default);

4.2 Procedure

The general procedure for extracting process models is illustrated in Figure 4.2.1. Each of the steps is described in more details below.

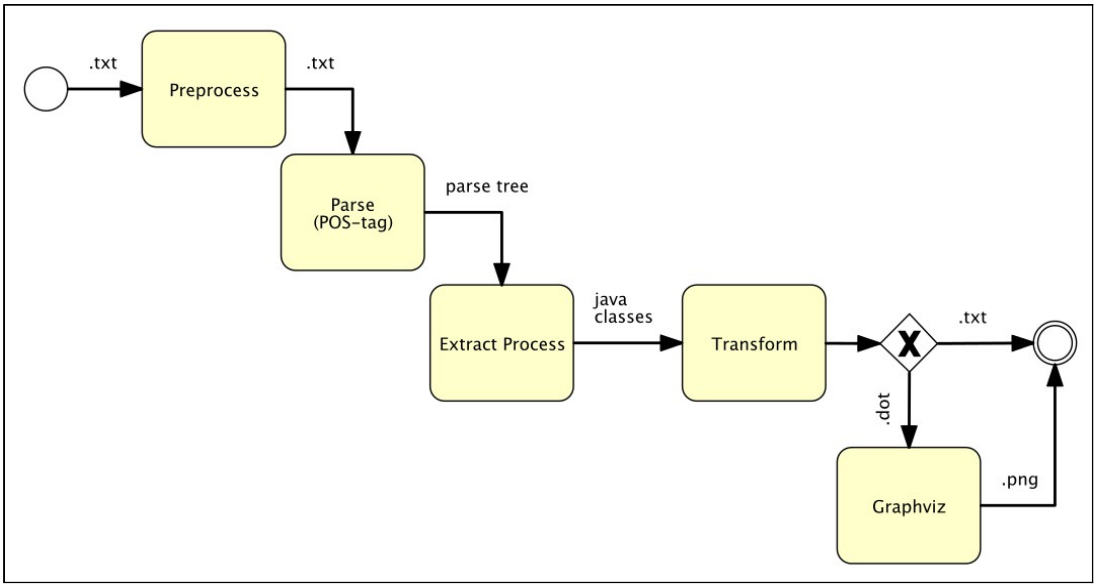


Figure 4.2.1. Flowchart.

1. **Preprocess**. Currently preprocessing includes only bringing all letters to the lower case. While this function can be extended to automated filtering of "noisy" sentences and structures.

   Input: text file.

   Output: text file.

2. **Parse (POS-tag)**. Parser is applied to parse the text and tag words with POS-tags.

   Input: text file.

   Output: parse tree with POS-tags.

3. **Extract process**. The main part of the algorithm where different filtering and matching rules applied in order to extract the process elements and relations between them

   Input: parse tree with POS-tags.

   Output: process model in java-class structure (the class structure is inherited from the one, described in Chapter 3, Figure 3.1.1.)

4. **Transform**. Transformation allows representation of a process model in different formats. Currently two formats are supported: textual mark-up and ".dot" format.

   Input: process model in java-class structure.

   Output: 1) free-format textual representation of a model;

   2) process model in ".dot" format.

5. **Graphviz**. Model in .dot format can be processed with Graphviz to get a process diagram in

a form of a graph.

Input: process model in .dot format.

Output: process diagram in a graphical format (e.g. “.png”, or “.gif”).

4.3 Components

The procedure was implemented in Java and is widely based on Stanford Parser together with Standford Tregex and Tsurgeon tree-manipulation tools.

Parser

Extraction rules for identifying process elements in a text are based on POS-tags, assigned to each of the words in a sentence. This assigning is done by a parser. The rules presented in Chapter 3 of this thesis were identified and validated using the POS-tags produced by Stanford Parser PCFG algorithm.

Stanford Parser is a state-of-the-art parser, written in Java by Stanford NLP group, open-source licensed under GPL - free to use for research purposes, free software projects, etc. Big advantage of using Stanford Parser is that it is well documented, supported and continuously updated (last update on May 2011 - Version 1.6.6: 2011-04-20). Staford Parser package contains 2 parsing algorithms: unlexicalized *PCFG* and lexicalized Factored parsers. Factored parsing algorithm runs 3 parsers including PCFG parser and includes lexicalization, hence it is considerably slower than PCFG. While the average precision of PCFG and Factored parsers is reported to be similar[25], that is why the decision was made in favor of PCFG.

The analysis, which was described in Chapter 3, revealed that not all of the sentences are parsed correctly. Some of the parsing errors directly cause errors in the subsequent process extraction phase. Hence, the performance of the whole process extraction procedure is limited by a performance of the corresponding parsing algorithm. The official precision of PCFG parser is estimated at ***86.36%*** [10].

Extraction rules

Rules for extracting patterns from narratives are based on the analysis documented in Chapter 3. Not all of the rules covered in the analysis, presented in Chapter 3, were implemented in the prototype.

The rules were implemented using Tregex and Tsurgeon tools. This syntax is based on tgrep syntax and is similar to regular expressions for extraction information from strings[24]. Tregex and Tsurgeon are tree-manipulation tools, also Java open-source from Stanford NLP Group. Tregex is used for identifying patterns in trees. Tsurgeon is based on Tregex patterns and performs operations on trees (e.g. delete, insert, move, adjoin).

Lemmatizer

Name of the activities, conventionally, have to be in Present Indefinite tense (base form or lemma of the verb). However verbs in natural language texts most of the time have other tenses, which produce “noise” in a process models, unless they are transformed into their base forms. During the process extraction phase lemmatisation method from Stanford CoreNLP Toolkit [25] is called in order to get the base form for a verb, which is used to name the corresponding activity.

5 Evaluation

5.1 Evaluation set

The program was evaluated on an industrial specification of an Internet billing process, the original of which can be found in Appendix 2. The process description was manually extracted from the corresponding table.

In order to make the text suitable for automated process model extraction, minimal preprocessing was required:

1. Omitted "." were inserted in the end of the sentences - because "." indicates end of a sentence for a parser ;
2. "." in "e.g." were removed – the same as for point 1;
3. ":" were transformed in "." - otherwise the marking of a type "Day 1:" is recognized as a part of a sentence - subject.

The preprocessed process description (Appendix 3) was used as an input for the prototype tool described in Chapter 4.

5.2 Evaluation results

The process model was automatically generated by the prototype in two output formats: mark-up text and as a graph diagram. As the diagram generation procedure is not mature enough to correctly represent complex compound structures, we recommend to refer to the process model, generated in textual format, to get the full and adequate representation of the extracted process elements. The full process model in the textual format can be found in Appendix 4, an extract from the generated process diagram is placed in Appendix 5.

The process diagram produced by the prototype was cross-validated against the one produced by Raven Cloud [23], which was previously described in Chapter 2. The produced models are too far from each other to be compared with numeric measures. The model produced by Raven is composed of disconnected process elements and most of the activities were not processed correctly.

In order to measure precision of the extracted model the following **method** was used. Each of the extracted process elements was assigned a score:

- 1 – element is identified correctly;
- 0,5 – element is identified correctly but has some minor errors;
- 0 – element was not identified correctly.

Errors produced by the lemmatizer (wrong stemming of the verbs) were ignored as not crucial ones. Also, since the proposed process extraction method is not able to identify "timer events" (which would be required in order to fully capture this process), all errors related to lack of timer events in the model were ignored.

The process description contains several phrases, which occur in the text at least twice. In order to exclude repeated errors from the analysis, two different validation sets were separated.

One of them contains elements extracted from all the sentences present in the text, while the other one contains only unique elements, ignoring duplicated clauses.

Precision and recall of the generated process model was evaluated separately for activities and control flows. Activities were evaluated together with their supporting elements – actors and artifacts.

All the errors discovered in the output process model are summarized and presented in Table 5.2.1. Due to the fact that some of these errors were caused by the parsing algorithm, they were analyzed separately from the errors produced by the process extraction method itself (the numbers in brackets correspond to the parsing errors). All parsing errors were analyzed and manually resolved by replacing the causes of the errors with their synonyms, which are parsed correctly (Cause and Quick Fix columns of the Table 5.2.1.). Therefore, the total number of errors and points for precision were calculated separately for the output, which includes parsing errors and the one produced after they were resolved. Precision and recall for the process models, which were produced before and after correction of the parsing errors, are presented in Table 5.2.2.

№	Error	#Unique	#Total	Points	Cause	Parsing Error	Quick Fix
1	Unsplit Activities	(2)	(5)	**(**-0.5**)**	“charged”	+	“charge you”
					missing comma between clauses		add coma
2	Unresolved Passive Voice	(1)	(2)	**(**-0.5**)**	“re-attempted”	+	“reattempted”
		1	1	-0.5	“have to”	-	-
3	Inanimate Actor	1	1	-0.5	“commence”	-	-
Total #Errors:		**2(5)**	**2(9)**				
Total #Points:		**-1(-2.5)**	**-1(-4.5)**				

Table 5.2.1. Errors in the extracted Activities.

#Activities:	#Unique	#Total
In the text	13	26
Identified	13(10)	26(15)
Identified Correctly	10.5(7.5)	21.5(10.5)
Precision	**0.81(0.75)**	**0.83(0.7)**
Recall	**1(0.77)**	**1(0.58)**

Table 5.2.2. Precision and recall for extracted Activities.

According to the tables presented above the process extraction method can be seriously harmed by the low precision of the underlying parsing algorithm. Hence, the ways to reduce the level of parsing errors should be more carefully studied, including application of another parsing algorithm or combining several of them to ensure adequate performance.

After elimination of all parsing errors, the control flow structure produced by the model was verified. The errors produced during the automated extraction control flows, are summarized in Table 5.2.3., the corresponding precision and recall are presented in Table 5.2.4.

№	Error	Control Flow	#Unique	#Total	Points	Cause	
1	Unidentified	Alternative	2	2	-	antonyms:	Customers with cash account/DDR account
							Credit Card/Direct Debit
2		Repetition	1	2		"**re**attempted"	
3	Unresolved reference	Sequence	1	2	-1	"DDR transactions that failed on Day 9"	
4		Repetition	1	3		"try again"	
Total #Errors:			**5**	**9**			
Total #Points:			**-2**	**-5**			

Table 5.2.3. Errors in the extracted Control Flows.

#Control Flows:	#Unique	#Total
In the text	10	25
Identified	7	21
Identified Correctly	5	16
Precision	**0.71**	**0.76**
Recall	**0.7**	**0.84**

Table 5.2.4. Precision and recall for extracted Control Flows.

All of the errors in extraction of the process elements, which were discovered during the validation of the implemented prototype, were covered by the analysis presented in Chapter 3. For example, the program is unable to identify alternative flows indicated by antonyms (customers with cash accounts/customer with DDR accounts; for Credit Card/for Direct Debit), unless it has a access to the ontology, where these antonyms are included. Also, the implemented prototype is not able to resolve references to previous activities. However, a repetition assigned through statements like "try again" is ambiguous, as it does not explicitly indicate which activity has to be repeated. In can refer to the previous activity, or to activity, which is previous to the current composite activity.

The scope of the flow is the most difficult element for automated identification. The prototype correctly identifies the start of alternative activities, signaled with "if" clauses. But, as "otherwise" is not explicitly indicated, the alternative, by default, becomes an option. The program is not able to identify the fact that if payment was already made, it will not be outstanding any more. In this particular example the problem can be resolved, by assuming that conditions with the same values are the same conditions, however again it is not always the case. Further in-depth research and analysis has to be performed in order to formulate and validate additional baselines for the method to be able to deal with ambiguity of a natural language.

All in all, the extracted process model is adequate enough to be used as a ready-made draft of the process model. A human analyst is able to correct the errors caused by the lack of semantic knowledge, e.g. resolve antonyms, inanimate actors and missing references.

6 Conclusion

This thesis proposed a method for semi-automated process model extraction from textual descriptions written in natural language (English). The method was designed according to the specific patterns and vocabulary typical for documents containing business process specifications. The extracted model consists of process elements (activities, actors, etc.) and relations between them. The model can be graphically represented as a diagram in different notations (e.g. BPMN, UML, Workflow nets, etc.)

A software prototype was implemented in order to validate the proposed method. The validation demonstrates a high precision and recall for activity extraction. However, lower performance was achieved when identifying "control flow relations" between activities. A step-by-step validation performed on a real case-study reveals the complexity of the problem, and highlights the strengths and limitations of the proposed method. All of the errors observed during the validation were predicted during the analysis phase.

As discussed in Chapter 5, our approach has to be extended in order to cover patterns for identifying not only the beginning of a control flow but also its scope. This problem may require additional in-depth analysis and development of methods for inferring composite activities from the text structure.

Another practical limitation of the proposed process model extraction method is its inability to detect and resolve anaphoras (e.g. pronouns). This limitation is significant due to the fact that anaphoras are very common in textual process descriptions. Addressing this limitation is therefore an obvious venue for future work.

Yet another practical limitation of the proposed method is its limited ability to separate the process-related information in a textual description, from contextual information. Typically a business document describing a business process will contain not only information about how the process is executed, but also contextual information related to the organizational context in which the process is executed. Also, business documents typically describe business rules that actors need to take into consideration when performing individual activities in the process. These business rules are usually not represented in the corresponding process diagram. In other words, a process diagram gives a filtered and abstracted view of a business document, but the proposed method has limited ability to perform this filtering, beyond filtering simple forms of "noise".

Many other limitations exist, some of which have been put into evidence during the evaluation. To a large extent, the contribution of this thesis has been to provide an initial approach to the problem of process model extraction, and to reveal the numerous inherent difficulties in dealing with process descriptions written in a natural language.

Resümee

Teema: Tekstipõhistest narratiividest protsessimudelite tuletamine

Lühikirjeldus: Minu töö eesmärgiks on luua metoodika, mille abil on võimalik ingliskeelsest tekstipõhisest protsessikirjeldusest tuletada äriprotsessi mudel. Sellel on praktiline tähtsus, kuna ärianalüütikud kavandavad protsessimudeleid tihti tekstilisest dokumentatsioonist. Kavandatava metoodika eesmärgiks on võimalikult suures mahus automatiseerida tekstist diagrammideks muutmise etappi. Loomulikud keeled on teatavasti väga keerukad ja mitmeti mõistetavad, seega selle projekti jaoks me läheneme probleemile kasutades "parima ürituse" meetodit, mis tähendab, et see meetod ei pruugi alati toimida. Kavandatav lähenemisviis suudab märgata teatud lause struktuure ja eraldada neist osalejad, tegevused ja objektid/artifaktid. Metoodika väljundiks on plokk-struktuuriga protsessimudel.

Metoodika rakendamiseks luuakse Java-põhine rakendus, mis baseerub vabavaralistele loomuliku keele töötlemise (NLP) teekidele. Täpsemalt, kõneosa (Part-of-Speech (POS)) sildistamine teostatakse kasutades Stanfordi parserit ja vastavalt POS siltidele, vastavate protsessi üksused määratakse kindlaks kasutades Tregex'i ja Tsurgeon'i. Praegune rakendus on juba võimeline tavapärastest lausetest tuvastama osalejaid, tegevusi/ülesandeid ja artifakte. Lisaks sellele on rakendus võimeline õigesti tõlgendama umbisikulise kõne konstruktsioone, vältima artikleid, sulge ning muid keerukaid keelestruktuure. On kavandatud, et rakendatakse ka Porteri tüve-tehnikat, et muuta tuvastus morfoloogilistele variatsioonidele vastupidavamaks. Rakenduse väljundiks on lihtne tekstiformaat esindamaks plokk-struktureeritud protsessimudeleid.

Oodatavad tulemused ja saavutused:

* Protsessimudeli kirjelduse analüüs, sealhulgas nimekiri lause struktuuride, ajaliste ja hargnemiste märkidest, mida tüüpiliselt tekstipõhistes narratiivides kasutatakse (näidates ära jadamisi, paralleelsed, alternatiivsed ja korduvtegevusvood);

* Java-põhine rakendus, mis on võimeline tuletama protsessimudeleid tekstilistest kirjeldustest "parima ürituse" meetodil

Abstract

Topic: Extraction of Process Models from Textual Narratives

Short description: The purpose of my work is to design a method to transform a textual process description (in English) into a business process model. This is of practical relevance, since process models are often designed by business analysts starting from textual documentation. The method to be designed aims at automating the text-to-diagram conversion phase as much as possible.

Natural languages are known to be highly complex and ambiguous. Accordingly, for this project we will approach the problem using a best-effort approach, meaning that the method is not intended to work always. Instead, the proposed approach will be able to detect certain sentence structures and extract actors, actions and objects/artifacts from them. Coordinating and subordinating conjunctions, as well as punctuation and other markers, will be used to identify sequencing, parallelism, conditional branching and repetition. The output of the method will be a block-structured process model.

The method is being implemented in Java based on open-source Natural-Language Processing (NLP) libraries. Specifically, Part-of-Speech (POS) tagging is performed using the Stanford parser and according to the POS tags, corresponding process entities are identified using Tregex and Tsurgeon. The current implementation is already able to identify actors, actions/tasks and artifacts from sentences that abide to certain common structures. Additionally the implementation is able to correctly interpret passive voice construction, avoid articles, parenthesis and other complex structures for the purpose of extracting essential information about the process. It is envisaged that a Porter stemming technique will be plugged into the implementation to make it more robust to morphological variations. Output is being produced in a simple text format for representing block-structured process models.

Expected results and achievements:

- analysis of process models' description, including a list of sentence structures, temporal and branching markers typically used in textual process narratives (indicating sequence, parallel, alternative and repetitive activity flows);
- a Java program capable of extracting process models from textual descriptions on a best-effort basis

Keywords: NLP, process extraction, business process sub-language.

References

1. Aditya Ghose, George Koliadis, Arthur Chueng: Rapid Business Process Discovery (R-BPD). In ER 2007, 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 2007, pp. 391-406.
2. Ann Bies, Mark Ferguson, Karen Katz, and Robert Mac-Intyre: Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania, 1995.
3. Artem Polyvyanyy, Luciano García-Bañuelos, Marlon Dumas: Structuring Acyclic Process Models. In BPM 2010, 8th International Conference on Business Process Management, Hoboken, NJ, USA, September 2010, pp. 276-293
4. Arthur H.M. ter Hofstede, W.M.P. van der Aalst, Michael Adams, Nick Russell. "Modern Business Process Automation: YAWL and its Support Environment", Springer, 2010. http://www.yawlbook.com/. Accessed on May 2011.
5. Attempto Project. *http://attempto.ifi.uzh.ch/site/* . Accessed on May 2011.
6. Avik Sinha, Amit M. Paradkar, Palani Kumanan, Branimir Boguraev: A linguistic analysis engine for natural language use case description and its application to dependability analysis in industrial use cases. In DSN 2009, IEEE/IFIP International Conference on Dependable Systems and Networks, Estoril, Lisbon, Portugal, June-July 2009, pp. 327-336.
7. Avik Sinha, Stanley M. Sutton Jr., Amit M. Paradkar: Text2Test: Automated Inspection of Natural Language Use Cases. In ICST 2010, 3rd International Conference on Software Testing, Verification and Validation, Paris, France, April 2010, pp. 155-164.
8. Bartek Kiepuszewski, Arthur H. M. ter Hofstede, Christoph Bussler: On Structured Workflow Modelling. In CAiSE 2000: 12th International Conference on Advanced Information Systems Engineering, Stockholm, Sweden, June 2000, pp. 431-445.
9. Colette Rolland, Camille Ben Achour: Guiding the Construction of Textual Use Case Specifications. In Data & Knowledge Engineering, Volume 25 (1-2), March 1998, pp. 125-160.
10. Dan Klein, Christopher D. Manning: Accurate Unlexicalized Parsing. In ACL 2003, 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, July 2003, pp. 423-430.
11. Graphviz – Graph Visualization Software. http://www.graphviz.org/. Accessed on May 2011.
12. J. Mike Jacka, Paulette J. Keller. "Business Process Mapping: Improving Customer Satisfaction", Wiley 2009.
13. Jan Mendling, Hajo A. Reijers, Wil M. P. van der Aalst: Seven process modeling guidelines (7PMG). In Information & Software Technology, Volume 52(2), January 2010, pp. 127-136.
14. Jan vom Brocke, Michael Rosemann (Eds.). Handbook on Business Process

Management 1. Introduction, Methods, and Information Systems. Series: International Handbooks on Information Systems, 2010, p. 131 (Help Desk Process).

15. Jaroslav Drazan, Vladimir Mencl: Improved Processing of Textual Use Cases: Deriving Behavior Specifications. In SOFSEM 2007, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 2007, pp. 856-868.
16. João Carlos de A. R. Gonçalves, Flávia Maria Santoro, Fernanda Araujo Baião: Business process mining from group stories. CSCWD 2009, 13th International Conference on Computers Supported Cooperative Work in Design, Santiago, Chile, April 2009, pp. 161-166.
17. João Carlos de A. R. Gonçalves, Flávia Maria Santoro, Fernanda Araujo Baião: Collaborative Business Process Elicitation through Group Storytelling. In ICEIS 2010, 12th International Conference on Enterprise Information Systems, Volume 3, ISAS, Funchal, Madeira, Portugal, June 2010, pp. 295-300.
18. João Carlos de A. R. Gonçalves, Flávia Maria Santoro, Fernanda Araujo Baião: Let Me Tell You a Story - On How to Build Process Models. In J. UCS 17(2011), Journal of Universal Computer Science , vol. 17, no. 2 (2011), pp. 276-295 .
19. Jussi Vanhatalo, Hagen Völzer, Frank Leymann: Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In ICSOC 2007, Fifth International Conference on Service-Oriented Computing, Vienna, Austria, September 2007, pp. 43-55.
20. Maribel Yasmina Santos, Ricardo Jorge Machado: On the Derivation of Class Diagrams from Use Cases and Logical Software Architectures. In ICSEA 2010, 5th International Conference on Software Engineering Advances , Nice, France, August 2010, pp. 107-113.
21. New South Wales. Independent Commission Against Corruption. Plant hire (heavy machinery): Corruption Prevention Project. Redfern, N.S.W. : 1992.
22. Oryx: Research. http://bpt.hpi.uni-potsdam.de/Oryx/Research. Accessed on May 2011.
23. RavenFlow Inc., RAVEN Cloud. http://www.ravencloud.com/. Accessed on May 2011.
24. Roger Levy, Galen Andrew: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In LREC 2006, 5th International Conference on Language Resources and Evaluation, Genoa, Italy, May 2006. http://ling.ucsd.edu/~rlevy/papers/levy_andrew_lrec2006.pdf. Accessed on May 2011.
25. The Stanford NLP (Natural Language Processing) Group. http://nlp.stanford.edu/software/index.shtml. Accessed on May 2011.

Appendix 1. Penn Treebank II Tags[9]

Clause Level

S - simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a *wh*-word and that does not exhibit subject-verb inversion.
SBAR - Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ - Direct question introduced by a *wh*-word or a *wh*-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.
SINV - Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ - Inverted yes/no question, or main clause of a *wh*-question, following the *wh*-phrase in SBARQ.

Phrase Level

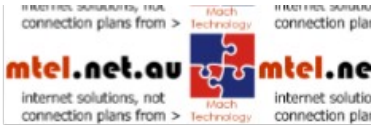
ADJP - Adjective Phrase.
ADVP - Adverb Phrase.
CONJP - Conjunction Phrase.
FRAG - Fragment.
INTJ - Interjection. Corresponds approximately to the part-of-speech tag UH.
LST - List marker. Includes surrounding punctuation.
NAC - Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP - Noun Phrase.
NX - Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to N-bar level but used quite differently.
PP - Prepositional Phrase.
PRN - Parenthetical.
PRT - Particle. Category for words that should be tagged RP.
QP - Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC - Reduced Relative Clause.
UCP - Unlike Coordinated Phrase.
VP - Vereb Phrase.
WHADJP - *Wh*-adjective Phrase. Adjectival phrase containing a *wh*-adverb, as in *how hot*.
WHAVP - *Wh*-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a *wh*-adverb such as *how* or *why*.
WHNP - *Wh*-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some *wh*-word, e.g. *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.
WHPP - *Wh*-prepositional Phrase. Prepositional phrase containing a *wh*-noun phrase (such as *of which* or *by whose authority*) that either introduces a PP gap or is contained by a WHNP.
X - Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing *the...the*-constructions.

9 From: Penn Treebank Tags. http://bulba.sdsu.edu/jeanette/thesis/PennTags.html. Accessed on May 2011.

Word level

CC - Coordinating conjunction
CD - Cardinal number
DT - Determiner
EX - Existential there
FW - Foreign word
IN - Preposition or subordinating conjunction
JJ - Adjective
JJR - Adjective, comparative
JJS - Adjective, superlative
LS - List item marker
MD - Modal
NN - Noun, singular or mass
NNS - Noun, plural
NNP - Proper noun, singular
NNPS - Proper noun, plural
PDT - Predeterminer
POS - Possessive ending
PRP - Personal pronoun
PRP$ - Possessive pronoun (prolog version PRP-S)
RB - Adverb
RBR - Adverb, comparative
RBS - Adverb, superlative
RP - Particle
SYM - Symbol
TO - to
UH - Interjection
VB - Verb, base form
VBD - Verb, past tense
VBG - Verb, gerund or present participle
VBN - Verb, past participle
VBP - Verb, non-3rd person singular present
VBZ - Verb, 3rd person singular present
WDT - Wh-determiner
WP - Wh-pronoun
WP$ - Possessive wh-pronoun (prolog version WP-S)
WRB - Wh-adverb

Appendix 2. Original Business Process Specification



Billing Process

Mach Technology Group Pty. Ltd.
ABN: 58 115 162 564

Shop 7 Homemaker Centre
18 Thomas Street, Noosaville QLD 4566

p: (07) 5474 3331
f: (07) 5474 3335
e: ask@mtel.net.au
w: mach.com.au

This document details important information about how your internet services account is billed. The day of the month your account is created represents 'Day 1' below (i.e. this may be any day of the month).

Please note: you must complete a new "Direct Debit Request Authority" form to avoid failed transaction charges if you:

- pay by credit card and you receive a new card, new expiry date or a lost card is replaced, etc
- change the banking details associated with your direct debit account.

Day of Month	Billing Procedure
Day 1:	Your invoice is emailed to you
Day 1-7:	Customers with cash accounts (i.e. no credit card or direct debit DDR authority) must ensure payment of the full outstanding amount is made and cleared in the Mach Technology bank account
Day 7:	Customers with DDR accounts (e.g. credit card or direct debit authority) have the full outstanding amount automatically deducted NB for Credit Card: if the automatic transaction fails for any reason, you will be notified by email, charged a $5.50 bank transaction failure fee and the system will try again on Day 9 NB for Direct Debit: if the automatic transaction fails for any reason, you will be notified by email, charged a $25 bank transaction failure fee and the system will try again on Day 9
Day 8:	If payment is still outstanding, you will be emailed a reminder
Day 9:	DDR transactions that failed on Day 7 will be re-attempted NB for Credit Card: if the automatic transaction fails for any reason, you will be notified by email, charged a $5.50 bank transaction failure fee and the system will try again on Day 11 NB for Direct Debit: if the automatic transaction fails for any reason, you will be notified by email, charged a $25 bank transaction failure fee and your Internet service will be suspended until payment is received by other means
Day 10:	If payment is still outstanding, an additional $10 late fee will be charged to your account
Day 11:	DDR transactions that failed on Day 9 will be re-attempted NB for Credit Card: if the automatic transaction fails for any reason, you will be notified by email, charged a $5.50 bank transaction failure fee and payment will have to be made by other means before Day 14
Day 14:	If payment is still outstanding, your Internet service will be suspended until payment is received (note Day 30 deadline below)
Day 30:	If payment is still outstanding, your account will be closed, disconnection fees will be applied and debt recovery proceedings will commence.

Mach Technology bank account details:
Direct Deposit payments should be made to:
- Bank: Westpac - Branch: Tewantin
- Acct Name: Mach Technology
- BSB No: 034-219 - Account No: 196074

Please note that DDR (i.e. Credit Card and Direct Debit) transactions will appear on your Bank Statement as:
ELECTRONIC PAYMENTS

Appendix 3. Preprocessed Process Description (Input)

Day 1. Your invoice is emailed to you.

Day 1-7.Customers with cash accounts must ensure payment of the full outstanding amount is made and cleared in the Mach Technology bank account.

Day 7.Customers with DDR accounts (eg credit card or direct debit authority) have the full outstanding amount automatically deducted.

NB for Credit Card. if the automatic transaction fails for any reason, you will be notified by email,
charge you a $5.50 bank transaction failure fee, and the system will try again on Day 9.
NB for Direct Debit. if the automatic transaction fails for any reason, you will be notified by email,
charge you a $25 bank transaction failure fee, and the system will try again on Day 9.

Day 8. If payment is still outstanding, you will be emailed a reminder.

Day 9. DDR transactions that failed on Day 7 will be reattempted.

NB for Credit Card. if the automatic transaction fails for any reason, you will be notified by email,
charge you a $5.50 bank transaction failure fee, and the system will try again on Day 11.
NB for Direct Debit. if the automatic transaction fails for any reason, you will be notified by email,
charge you a $25 bank transaction failure fee, and your Internet service will be suspended until payment is received by other means.

Day 10. If payment is still outstanding, an additional $10 late fee will be charged to your account .

Day 11.DDR transactions that failed on Day 9 will be reattempted.

NB for Credit Card. if the automatic transaction fails for any reason, you will be notified by email,
charge you a $5.50 bank transaction failure fee, and payment will have to be made by other means before Day 14.

Day 14. If payment is still outstanding, your Internet service will be suspended until payment is received (note Day 30 deadline below).

Day 30. If payment is still outstanding, your account will be closed, disconnection fees will be applied and debt recovery proceedings will commence.

Appendix 4. Process Model in Textual Format (Output 1)

"BILLING" PROCESS

<SEQUENCE\>

Activity: email your invoice to you
Artifact: your invoice
Artifact: you

Actor: customers with cash accounts
Activity: ensure made payment of full outstanding amount and cleared in mach technology bank account
Artifact: payment
Artifact: full outstanding amount
Artifact: mach technology bank account

Actor: customers with ddr accounts
Activity: have full outstanding amount automatically deduct
Artifact: full outstanding amount

<ALTERNATIVE\>
Clause: automatic transaction fails for any reason

<SEQUENCE\>

Activity: notify you by email
Artifact: you
Artifact: email

Activity: charge you $ 5.50 bank transaction failure fee
Artifact: you
Artifact: $ 5.50
Artifact: failure fee

<REPETITION\>
Clause: again

Actor: system
Activity: try on day 9
Artifact: day 9

</REPETITION>

</SEQUENCE>

</ALTERNATIVE>

<ALTERNATIVE\>
Clause: automatic transaction fails for any reason

<SEQUENCE\>

Activity: notify you by email
Artifact: you
Artifact: email

Activity: charge you $ 25 bank transaction failure fee
Artifact: you
Artifact: $ 25 bank transaction failure fee

<REPETITION\>
Clause: again

Actor: system
Activity: try on day 9
Artifact: day 9

</REPETITION>

</SEQUENCE>

</ALTERNATIVE>

<ALTERNATIVE\>
Clause: payment is still outstanding

Activity: email you reminder
Artifact: you
Artifact: reminder

</ALTERNATIVE>

Activity: reattempted ddr transactions that fail on day 7
Artifact: ddr transactions
Artifact: day 7

<ALTERNATIVE\>
Clause: automatic transaction fails for any reason

<SEQUENCE\>

Activity: notify you by email
Artifact: you
Artifact: email

Activity: charge you $ 5.50 bank transaction failure fee
Artifact: you
Artifact: $ 5.50
Artifact: failure fee

<REPETITION\>
Clause: again

Actor: system
Activity: try on day 11
Artifact: day 11

</REPETITION>

</SEQUENCE>

</ALTERNATIVE>

<ALTERNATIVE\>
Clause: automatic transaction fails for any reason

<SEQUENCE\>

Activity: notify you by email
Artifact: you
Artifact: email

Activity: charge you $ 25 bank transaction failure fee
Artifact: you
Artifact: $ 25 bank transaction failure fee

<ALTERNATIVE\>
Clause: until received payment by other means

Activity: suspend your internet service
Artifact: your internet service

</ALTERNATIVE>

</SEQUENCE>

</ALTERNATIVE>

<ALTERNATIVE\>
Clause: payment is still outstanding

Activity: charge additional $ 10 late fee to your account
Artifact: additional $ 10 late fee
Artifact: your account

</ALTERNATIVE>

Activity: reattempted ddr transactions that fail on day 9
Artifact: ddr transactions
Artifact: day 9

<ALTERNATIVE\>
Clause: automatic transaction fails for any reason
<SEQUENCE\>

Activity: notify you by email
Artifact: you
Artifact: email

Activity: charge you $ 5.50 bank transaction failure fee
Artifact: you
Artifact: $ 5.50
Artifact: failure fee

Actor: payment
Activity: have to be make by other means before day 14
Artifact: other means
Artifact: day 14

</SEQUENCE>

</ALTERNATIVE>

<ALTERNATIVE\>
Clause: payment is still outstanding

<ALTERNATIVE\>
Clause: until received payment

Activity: suspend your internet service
Artifact: your internet service

</ALTERNATIVE>
</ALTERNATIVE>

<ALTERNATIVE\>
Clause: payment is still outstanding

<SEQUENCE\>

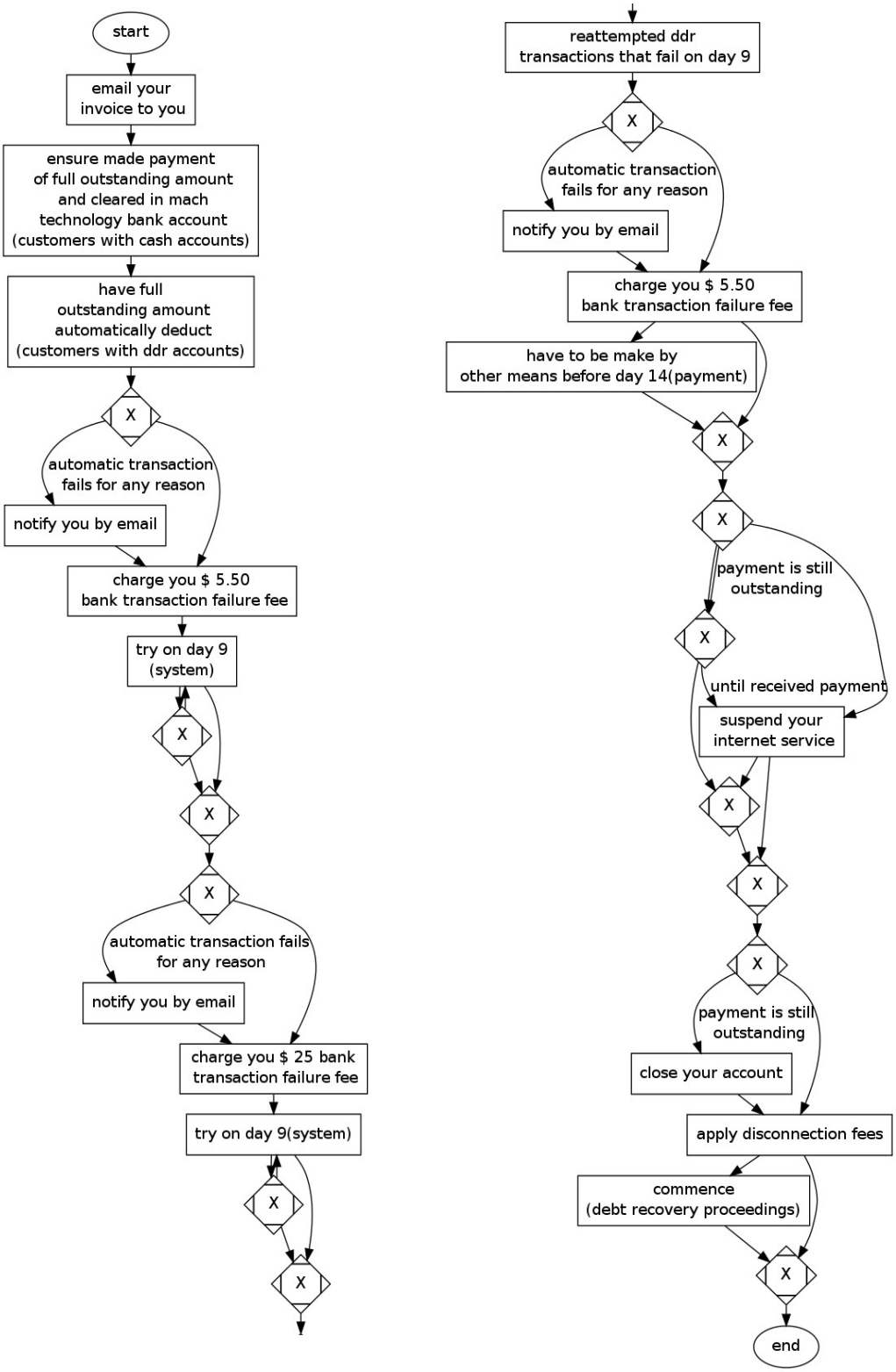
Activity: close your account
Artifact: your account

Activity: apply disconnection fees
Artifact: disconnection fees

Actor: debt recovery proceedings
Activity: commence

</SEQUENCE>
</ALTERNATIVE>
</SEQUENCE>

Appendix 5. Extracts from the Process Diagram (Output 2)



Appendix 6. Process Diagram by Raven Cloud