2014•2015
FACULTY OF BUSINESS ECONOMICS
Master of Management

Master's thesis

Comprehension of Business Process Models: An evaluation of metrics for understandability

Supervisor :
Prof. dr. Benoit DEPAIRE

Koen Van Eijk

Thesis presented in fulfillment of the requirements for the degree of Master of Management

universiteit hasselt
KNOWLEDGE IN ACTION

Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE-3500 Hasselt
Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE-3590 Diepenbeek

universiteit hasselt
KNOWLEDGE IN ACTION

2014•2015
FACULTY OF BUSINESS ECONOMICS
Master of Management

Master's thesis

Comprehension of Business Process Models: An evaluation of metrics for understandability

Supervisor :
Prof. dr. Benoit DEPAIRE

Koen Van Eijk

Thesis presented in fulfillment of the requirements for the degree of Master of Management

universiteit hasselt
KNOWLEDGE IN ACTION

Preface

Writing a master's thesis is not a simple assignment: it requires a lot of effort, motivation and responsibility. I am therefore very thankful for the support of the people who I would like to thank in this preface.

Firstly, I want to thank my promoter, prof. dr. Depaire, for his support in the writing of this master's thesis in general. It was his valuable feedback and flexibility that made the writing of this research project possible.

Secondly, I want to thank my parents for giving me the opportunity to pursue an academic degree. It is thanks to their motivational, emotional and financial support that made me what I am today.

Lastly, I would like to thank my girlfriend for her loving support and for proofreading my writings.

I sincerely hope this master's thesis can be as instructive and interesting for the reader as it was for me writing it.

Koen van Eijk

Diepenbeek, June 2015

Summary

For a business it is important that all stakeholders can participate throughout all the phases of business process design. Not only the way the business is currently working, but also the way it is supposed to work are important topics that need to be communicated thoroughly between all parties involved and this underlines the importance of understandability of business process models (BPM). Understandability can be linked to the complexity of BPM. Complexity can be seen as *cognitive complexity* (for humans) or structural complexity (for computers). This master's thesis focusses on the cognitive complexity of BPM.

The literature concerning BPM understandability has already adapted numerous metrics from software engineering research. However, these metrics seem to measure different aspects of cognitive and structural complexity. This master's thesis tries to identify useful combinations of metrics for assessing BPM understandability. Better metrics should lead to better conclusions, which ultimately should result in better communication of BPM.

Firstly, a literature review is conducted. Popular theories from cognitive sciences are identified which can be linked to BPM understandability. Furthermore, different aspects of both model and user are discussed that might influence the understandability according to different studies. In the literature review some popular complexity metrics are discussed which could serve as a proxy for BPM understandability. Understanding of BPM is decomposed in the syntax, semantics and pragmatics level.

Secondly, Van der Aalst's Workflow Patterns are used as a basis for comparing the different complexity metrics. The patterns are expressed using the BPMN notation and the complexity level for the different metrics is computed. By the use of mean weighted values and rankings it was found that the different metrics are not equally sensitive. They do however (overall) give the same ranking to the patterns with regards to their complexity. By applying factor analysis on the data it was found that the metrics address two different aspects of complexity. On the one side some metrics address structural complexity and on the other side some metrics address functional complexity. It is concluded that it is preferable to use combinations of metrics that address these two different aspects.

Some guidelines are given for future research. Not much research has apparently been done yet on the secondary notation of BPM (layout, colors, direction, …). BPMN 2.0 has capabilities of expressing

actor responsibility and accountability, and what resources are used to complete tasks. How these aspects relate to the comprehensibility of BPM has supposedly not been addressed yet. Lastly, in this master's thesis only Workflow Patterns have been discussed, which limits the research to the syntax level. To my knowledge only limited research has been done for the two following levels, semantics and pragmatics, which could be an interesting starting point for future research.

Table of contents

List of figures

List of tables

List of abbreviations

BPM: Business Process Models, 1
BPMN: Business Process Models and Notation, 3
BSC: Balanced scorecard, 27
CFC: Control Flow Complexity (metric), 22
CTML: Cognitive Theory of Multimedia Learning, 8
CW: Cognitive Weights, 24
DPP: Domain Process Patterns, 32
EPC: Event Driven Process Chains, 1
IC: Interface Complexity (metric), 26
IF: Information Flow (metric), 26
IS: information systems, 4
LOC: Lines of Code (metric), 18
MaxND: Maximum Nesting Depth (metric), 26
MeanND: Mean Nesting Depth (metric), 25
NOA: Number of Activities (NOA), 18
NOAC: Number of Activities and Control flows (metric), 18
NOAJS: Number of Activities and Joins and Splits (metric), 18
UML: Unified Modeling Language, 1
UML AD: Unified Modeling Language Activity Diagrams, 25
WFP: Workflow Patterns, 32
YAWL: Yet Another Workflow Language, 25

Chapter 1: Introduction

1.1 Problem description

Within the research field of process modeling, complexity of business process models (BPM[1]) is a criterion that has been a popular target of study. Process complexity, within the context of process modeling, can be regarded in its simplest form as the amount of nodes and arcs in the graphical representation of a model (Wil M.P. van der Aalst, 2011). However, there are two different sides to the coin when it comes to BPM complexity. On the one side there is the level of complexity in a mathematical sense (for computers) and on the other side the complexity for human beings. In academic literature, the former is often referred to as *structural complexity*, the latter is referred to as *cognitive complexity* (Cruz-Lemus, Maes, Genero, Poels, & Piattini, 2010).

Keeping *grip* on an acceptable level of process complexity is of great importance to keep the model understandable and in turn usable. If a model cannot be understood, what is the point of generating a model in the first place? It is therefore not only important to assess the level of necessary complexity to select the appropriate model, but also selecting the right language in which the model is expressed to its users (Nordbotten & Crosby, 1999). The users might not be entirely familiar with the language and might interpret the BPM differently, which may cause confusion or unintended behavior (Eriksson & Penker, 1998). The notational language is only one of the many design choices that the designer of a BPM can make. For a business it is important that all stakeholders can participate throughout all the phases of business process design, whether it is to communicate how the business is currently working (as-is modeling) or how it is supposed to work in the future (to-be modeling) (Burton-Jones & Meso, 2008). Therefore in a logical sense it is of uttermost importance to keep the business processes understandable, not only to business process model specialists but also with regards to novices or people who are not confronted with them on a regular basis.

There are already many different modelling languages available today (List & Korherr, 2006): UML 2.0 Activity Diagrams, Business Process Modeling Notation, Event Driven Process Chain (EPC) (Scheer, 2000), Petri Nets (Murata, 1989) … Each language has its user base and followers because they have different origins and are aimed at different domains. This study however tries to identify properties that transcend the language in which the BPM is expressed.

[1] The abbreviation 'BPM' is used for business process models throughout the rest of the text to increase readability.

In order to improve communication of BPM, this study is an evaluation of different available metrics for assessing process model understandability. These metrics are applied to Van der Aalst's Workflow Patterns (2003) as a basis for evaluation.

1.2 Research question

The general research question of this master's thesis is:

"What are useful metrics or combinations of metrics for assessing business process model understandability?"

To find an answer to this general research question, it is divided into smaller questions:

- Are all metrics equally sensitive?
- Do the metrics give similar ranking to patterns?
- Do the metrics measure the same thing?
- What patterns are more complex? Why could this be the case?

1.3 Research design

This master's thesis consists of two major sections. One section being a literature review, which consists of an exploration within the field of cognitive process model complexity and process model understandability based on theory and metrics. The literature review is an important prerequisite for this master's thesis. It should offer the opportunity to gain insights from past research which allows to build conclusions on well-grounded theory. Therefore the preference is to employ resources that are peer-reviewed and enjoy high reliability in the academic world. One specific goal of the literature review is to find out what past research has already contributed to the domain of process model understandability. This is done to get an overview of what has, and more importantly what has not, been addressed so far in academic literature.

The other section tests the metrics discussed in the literature review at the syntactical level of Van der Aalst's Workflow Patterns (2003). The different patterns are compared with regards to their measured complexity and what the implications might be for the understandability of the usage of such patterns in BPM. Then another comparison is made at the metric-level: what metrics might be useful for estimating cognitive complexity of BPM and which combinations of metrics make sense and which do not.

1.3.1 Goals

A lot of research has already been done on the understandability of BPM in general, and some notations in particular. In the past, evidence has been found of significant variation in model interpretation when different graphic styles are used (Nordbotten & Crosby, 1999). For a business, changing 'the way things are done' with regards to process modeling can be a costly enterprise. Improving the communication of business process models is a win-win situation for all parties. If the business process is understood in a better way, the transition from process understanding to process improvement will be easier.

1.3.2 Methodology

To find interesting literature, search engines like Google Scholar and EBSCOhost were used. Examples of search terms used were: *process model understandability, process model complexity, cognitive complexity process model, understanding process models, business process models, business process modeling, BPM understanding, BPM complexity, …* The snowball method is applied in both directions. Interesting studies can lead to interesting references, but Google Scholar is used to find more recent studies that refer to the literature that is found. More recent studies might lead to newer and better conclusions with better testing methods.

Popular metrics for BPM complexity are identified, discussed and tested for Van der Aalst's Workflow Patterns which are translated in BPMN 2.0. A comparison is then made on the level of the patterns (which patterns are more complex?) and another comparison is made on the level of the metrics. This should allow us to answer the research questions formulated above.

When graphical representations of BPM are given, the preference is given to the BPMN 2.0 notation as it is the most popular notation for business process modelling. These examples are designed in the Bizagi Modeler[2], part of the Bizagi software suite.

1.3.3 Scope

The entire research is conducted with a business context in mind. The conclusions of this master's thesis should contribute to the pool of knowledge that is readily available to be applied in practice and propose a framework that can be used as a foundation for further research.

[2] More information on the Bizagi software suite can be found on http://bizagi.com/

Applicability of research results

The conclusions drawn from this master's thesis could be used in practice to evaluate the understandability of BPM and to more easily identify and tackle the factors that negatively influence it. This allows businesses to increase the comprehension and allow better communication of BPM, which are frequently used to communicate various stages in information systems (IS) design. Applicable situations can be systems analysis, communication design, organizational re-engineering, project management, end-user querying and many others (Recker, Rosemann, Indulska, & Green, 2009).

If more time (money) is saved in the understanding stage of a BPM, more time (money) can be spent on the subsequent stages (business process improvement, restructuring, …). This makes any study in the field of BPM which contributes to the creation of higher quality models indeed relevant (Davies, Green, Rosemann, Indulska, & Gallo, 2006).

Compared to research on complexity in software engineering, the amount of research done in the field of BPM complexity is still limited despite its importance. This is why any further research in this field can be considered useful.

Chapter 2: Theories on understandability

2.1 Why complexity matters

In the discovery phase of process model mining, it has been suggested that there are certain *quality criteria* which greatly affect the usability and understandability of the end result: fitness, generalization, precision and last but not least simplicity (Wil M.P. van der Aalst, 2011). According to Van der Aalst it is hard to find the right balance between these dimensions when conducting process discovery. He states that increasing simplicity (or lowering complexity) might lead to lower fitness or reduced precision. *Fitness* being the level of 'fit' that the model has with the traces that have been used in the discovery phase. He also states that *over-fitting* could lead to overly complex process models which are harder to understand. In turn, this might lower usability within a business management context.

2.1.1 How complexity relates to understandability

Communicating a 'simplified' version of a process model might be easier than a more complex interpretation of the same model. One might suggest that the greatest importance in evaluating between these quality criteria is the purpose of the model. If the actual purpose of the model is to check for nonconformity or deviations in a risk-aversive setting, then every trace within the model is of equal importance and should not be left ignored. One deviation in a business context might be enough to indicate that the internal control of a process has failed which may possibly have led to theft or fraud concerning company assets (Rozinat & van der Aalst, 2008). The particular deviation probably should lead to further investigation on the level of the specific case.

However, if the model is intended to be used for communicating an overall view of a business process between disciplines in a company, not all deviations should be considered as important. There is a certain danger of over-fitting the model with the trace log data which might lead to what Van der Aalst refers to as the extreme of the so-called *flower model*. In this type of model every trace in the log can be *played out* perfectly, however it does not have any useful implications with regards to generalizability: everything is still possible within the model (Wil M.P. van der Aalst, 2011). This unnecessarily complicates the model which in turn makes it harder to understand and communicate in a business context (Figl & Laue, 2011).

2.2 Defining understandability

In order to define the concept of *understandability*, one could adapt Biggerstaf's definition within the context of a software program:

> *"A person **understands** a BPM when they are **able to explain** the BPM, its **structure**, its **behavior**, its **effects on its operational context**, and its **relationships on its application domain** in terms that are qualitatively different from the tokens used to construct the BPM in a modeling language."*
>
> *(Biggerstaff, Mitbander, & Webster, 1994)*

This definition can be linked to semiotic theory (Burton-Jones, Wand, & Weber, 2009) and the SEQUAL model (Lindland, Sindre, & Solvberg, 1994), as well as Cognitive Load Theory (Sweller & Chandler, 1994). These concepts with the addition of Resource Allocation Theory (Kanfer, Ackerman, Murtha, Dugdale, & Nelson, 1994) and the Multimedia Theory of Learning (Mayer, 2002) will be further discussed below. These theories, often borrowed from cognitive sciences, are popular target of reference for the field of study in BPM complexity and understandability. Firstly, every theory is shortly summarized and explained. Secondly, the link is made with BPM understandability.

2.2.1 Cognitive Load Theory

A popular assumption to make the conceptualization of comprehensibility more tangible, is that a BPM is harder to understand as the number of elements in the model increases. This assumption is grounded by Cognitive Load Theory (Sweller & Chandler, 1994). According to this theory, comprehension is negatively affected if the amount of information that needs processing exceeds the working memory (Paas & Ayres, 2014). Cognitive Load Theory distinguishes between two different kinds of memory, namely the *limited short-term working memory* and the *unlimited long-term memory*. The short-term memory is limited in such a way that it only allows for (approximately) seven items of information at a given time. This theory has implications in what ways a user will have difficulty in understanding (complex) BPM. When the user is confronted with new theory (for instance a BPM describing a business process) this generates a load on his working memory. If he cannot fall back on long-term memory (experience, theoretical knowledge, …) this burden will be greater, according to the theory (Sweller & Chandler, 1994). Following this logic, someone with

extensive knowledge and/or experience (expertise) will have an easier time understanding a BPM than someone who is a novice with the practice.

Implications for BPM understandability

Cognitive Load Theory implies that when more elements are added to the BPM, it should become more difficult to understand. Experimental studies in the past have already shown that increased model size and complexity affect understanding of the model (Aguilar, García, Ruiz, & Piattini, 2007; H. A. Reijers & Mendling, 2011). It has been shown by Mendling e.a. (2012) that for example real activity labels decrease the syntactical process model understanding. This means that if labels with 'meaning' are added rather than abstract letters or numbers the syntactical understandability of the model decreases. The user has a harder time understanding the syntax of the BPM when labels with significant meaning are added. For application in practice this means that if the designer of a process model wants to check the syntactical logic of his BPM he should make abstraction of the labels. This would 'free up memory' by eliminating the labels of nodes and arcs which in turn should make the analysis of the syntax of the model easier (Mendling e.a., 2012).

2.2.2 The SEQUAL model and Semiotic Theory

The SEQUAL model (Lindland e.a., 1994) proposes three levels of model quality, namely the *syntactic*, *semantic* and *pragmatic* level. The syntactic level of a BPM can be seen as the language that is used for the graphical notation of the process. These are the various nodes and arcs showing the control flow of the process. When, for instance, labels and icons are added, the semantic level is reached. The nodes and arcs gain semantic meaning because they demonstrate real activities and their decision logic. Lastly, if the context is added (be it a business context) the pragmatic level is reached and the process model can represent knowledge for action (Krogstie, Sindre, & Jørgensen, 2006). According to semiotic theory, understanding precedes communicating (Burton-Jones e.a., 2009). This relationship can be seen as a 'ladder': before semantics can be understood, the user firstly has to understand the syntax of the model. When both syntax and semantics are understood, then operational context can come in to support the pragmatic level of understanding.

Implications for BPM understandability

When applied to BPM, the syntax can be seen as the graphical representation of arcs and nodes. When labels are added, the model is elevated to the semantic level. It gives logical meaning to the different arcs and nodes. When the context is then added, for example a production plant striving for cost leadership, the model is elevated to the pragmatic level (Mendling e.a., 2012). This 'ladder'

of Burton-Jones e.a. (2009) also gives us the reason why BPM understandability is so important: if syntax and semantics are not understood, knowledge for action cannot be attained.

2.2.3 Resource Allocation Theory

According to resource allocation theory, understanding something you have experience with is easier than without any past experience with the subject (Kanfer e.a., 1994). This is due to the fact that the demand for cognitive attention is reduced which results in freed up cognitive resources which in turn can be used to improve task production or outcome production (Kanfer e.a., 1994). This allows someone with a higher amount of experience to more easily understand complex concepts than someone with lesser experience. This is due to the fact that the person with lesser experience does not have a long term memory to fall back on and therefore has to *allocate* all his mental *resources* to understanding the concept.

Implications for BPM understandability

In terms of understanding BPM this would imply that the user can understand the model better and faster if he has experience with BPM: he can assign all his cognitive resources on actually understanding or improving the process instead of understanding just the syntax or semantics of the BPM. According to experiments by Mendling e.a. (2012) theoretical knowledge and process modeling expertise are important factors on the formal comprehension of process models. The 'task', to use Kanfer's terminology, is understanding the BPM. The 'outcome' is whether or not the user acts as is instructed by the BPM, assuming voluntariness. In his research, Mendling therefore stresses the need for education and introduction to the usage and advantages of BPM to increase the understanding of more complex models.

2.2.4 Cognitive Theory of Multimedia Learning

Continuing from the 'ladder' of semiotic theory, going from solely *syntactic* factors influencing the understandability of a BPM to the *semantic* level, one could apply the Cognitive Theory of Multimedia Learning (CTML) (Mayer, 2002). The CTML suggests that there are three actors involved when it comes to understanding explanative information:

- The content or the message (the business process)
- The presentation: the way the content is presented (notational aspects, …)
- The user: the personal characteristics of the viewer

The actual level of understanding achieved afterwards is also divided into three groups:

- No understanding at all
- Surface understanding
- Deep understanding

These levels are achieved by the product of two variables as defined by Mayer: *retention* and *transfer*. Retention is defined as understanding the learning material, and transfer is defined as the ability to use the deeper understanding gained from the material to apply it in different situations. No understanding is achieved when both of these elements are low. Surface understanding is achieved when retention is high, but transfer is low. This means that the information is received but does not find integration with knowledge gained in the past. When both are high, the level of deep understanding is attained and the knowledge is integrated in the long-term memory where it can be accessed to solve new problems (Burton-Jones & Meso, 2008).

Implications for BPM understandability

CTML has a lot of implications for the understandability of BPM. For starters, the different actors can influence the level of cognitive complexity of the BPM. Firstly, the content that is being communicated as learning material is the business process which the BPM represents. If the business process is complex, the model that has to communicate this process will most likely be complex as well. Secondly, the presentation style has influence on the level of understanding according to CTML. This means that the way in which the BPM is presented has an important role in the level of understanding the user will achieve. Lastly, the individual characteristics of the user have their impact on the understandability of the BPM. If the user has more experience working with BPM, he will most likely understand it more easily. The levels of understanding as defined by Mayer (2002) can be linked with the syntactical, semantic and pragmatic understanding. The user needs to understand the syntax and semantics to achieve surface understanding, but to reach deep understanding he needs to understand the BPM at the pragmatic level.

2.3 Levels of understanding

The three levels of understanding are based on the SEQUAL model by Lindland e.a. (1994) and semiotic theory (Burton-Jones e.a., 2009) as discussed earlier. Here they are assumed as a ladder with levels rather than independent dimensions. The user firstly has to understand the syntax before he can understand the semantics. When the user understands the semantics of the model he can achieve 'deep' understanding and reach the pragmatics level (Mayer, 2002). These relationships are rather straightforward and the analogy can be made with a story. If you cannot read the letters and

words (*syntax*), you cannot understand the plot (*semantics*). If you cannot understand the plot, you cannot understand the underlying message intended by the author of the story (*pragmatics*). For BPM specifically, this means that the syntactical aspect of the model is of greatest importance. If the syntax is misunderstood, the interpretations of the user of the model are most likely flawed as well (H. A. Reijers & Mendling, 2011; H. Reijers & Mendling, 2008). This however does not imply that the other levels should be neglected: their mere presence is crucial for the good understanding of BPM. This necessity can be illustrated by the fact that a BPM does not have much meaning without labels (semantics) and without the business environment context (pragmatics).

2.3.1 Syntax

The syntax of a BPM is the graphical notation (language) it is written in and the logic that is expressed by the way activities and gateways interact with each other through control flows (Mendling e.a., 2012). As discussed before, the user firstly has to understand the syntax of the BPM before he can get to the other different levels of understanding. This means that the user must be somewhat familiar with the notational language and its logic and that the designer of the BPM should avoid making syntactical errors (Figl, Mendling, & Strembeck, 2013) and avoid the use of anti-patterns (Laue & Awad, 2010). Anti-patterns are specific patterns that are known to be harder to understand or have other negative consequences. The term originates from software design research (Gustafsson, 2000), however it can be adapted to BPM. The detection of these anti-patterns can be an indication of poor BPM design and understandability (Gruhn & Laue, 2006b). Use of the anti-pattern might seem logical at first sight, but the disadvantages outweigh the advantages in the long run. Some examples of these anti-patterns are discussed in a paper of Laue and Awad (2010). The importance of syntax is partly demonstrated in the amount of research that has been done into syntactical comprehension with regards to BPM and metrics that measure it (Figl & Laue, 2011; Genon, Heymans, & Amyot, 2011; Gruhn & Laue, 2009; H. A. Reijers & Mendling, 2011).

2.3.2 Semantics

Semantics are the meaning of the activities. The semantic level of the BPM is reached when labels are added to the various activities and gateways (Mendling e.a., 2012). Mendling e.a. conclude their research into activity labels that in the designing phase of a BPM it can be useful to temporarily hide or abstract the activity labels to evaluate the BPM for syntax errors. They found that on the syntactical level users could more easily comprehend the BPM when activity labels were abstracted as letters (A, B, C, …). A BPM is however more than simply syntax logic, it needs meaning as well (*semantics*).

2.3.3 Pragmatics

When context is introduced to the BPM, the pragmatics level is reached. So far not much knowledge has been generated by the literature with regards to the pragmatic factors of business process modelling (Burton-Jones e.a., 2009). The reason for this can probably be found in the fact that the preceding levels (at least according to semiotic theory) have not been studied thoroughly yet.

2.4 Factors influencing understandability

One important aspect of models is that they should be easily understandable and that they are intuitive (Davies e.a., 2006). Apart from the obvious positive relationship between size and complexity of business models, this paragraph describes other phenomena that might have their impact on the understandability of BPM. In this master thesis we employ the definition for BPM understandability as given by Reijers and Mendling (2011): "the degree to which information contained in a business process model can be easily understood by a reader of that model". The literature demonstrates different points of view with regards to BPM understandability. This paragraph attempts to harmonize these views which should result in a framework that can be used as a cognitive 'map' for BPM understandability.

Figl and Laue (2011) identified three relevant factors that influence the understandability of business process models: *relations between elements*, *element interactivity* and *element separateness*. The *relations between elements* refer to the different control structures within the BPM. Some structures might be harder to understand (order, concurrency, repetition and exclusiveness) than others (Melcher, Mendling, Reijers, & Seese, 2010). The *element interactivity* is defined as the way the elements interact with each other and whether they occur in serial or parallel. The higher the parallelism of these interacting elements, the higher the cognitive load for the reader of the BPM (Figl & Laue, 2011). Figl and Laue measure this by computing the distance of the elements in a process structure tree. The higher this distance, the more complex the interactions between elements are to understand for the model reader. Lastly, *element separateness* is addressed as the amount of cut-vertices in the BPM. The further two elements are removed from each other, the more easily the model reader creates reference points and he can more easily understand the model (Mendling & Strembeck, 2008).

Reijers and Mendling (2011) apply a cognitive dimensions framework that has been empirically confirmed for the understandability of software and visual notations (Green & Petre, 1996). The

cognitive dimensions they apply to BPM are: *abstraction gradient*, *hard mental operations*, *hidden dependencies* and *secondary notation*. With *abstraction gradient* the framework refers to the capability that a notation can handle the grouping of elements. This means that if the model gets more complex it becomes harder to understand because it is harder to identify relations within the BPM. Reijers and Mendling (2011) presume that this is why expert modelers are better at finding related parts within the BPM, and thus understand it better than a novice would. The *hard-mental operations* refer to the fact that BPM become over-proportionally harder to understand as their size increases. Again, according to Reijers and Mendling (2011), experts should be more capable of decomposing the model in smaller chunks to deal with size issues. The *hidden dependencies* are the dependencies between the BPM elements that are not directly visible. Reijers and Mendling associate these hidden dependencies with the relations between split and join connectors in BPM. They propose that expert modelers make use of heuristics to easily understand a BPM's behavior. Lastly, the *secondary notation* is the part of the BPM that is not formalized by a standard. For example, BPMN 2.0 does not define standards related to the empty space between elements, or how long flow connectors should be. This extra information in the BPM is not part of the formal notation, but is added by the designer of the model. Intentionally, this can be done through applying labeling conventions as addressed by Mendling e.a. (2010) or by following specific layout strategies as discussed by Ware e.a. (2002). Mendling and Reijers (2011), next to model factors, also address the importance of personal factors with regards to BPM understandability. They identify expertise and personality as the two most important personal factors. They conclude their research that expertise can overcome even the poorest designed BPM. They propose that it might be more interesting for companies to invest in training staff rather than put effort in reducing the cognitive complexity of their process models.

Schrepfer e.a. (2009) distinguish two important aspects to the process of reading and understanding BPM. Firstly they identify the aspect of *graphical readership*. This describes the user's ability to read a BPM. This means that the user has to see the graphical elements of the model and interpret their meaning (Petre, 1995). This part of the process with regards to BPM is not much impacted by intuition, but it is closely linked to the notation that is used to express the model. It is therefore of great importance to think about the notation and layout style employed in the BPM. Secondly they identify the aspect of pattern *recognition*. Using patterns much like described in Van der Aalst's Workflow Patterns (2003) allows the user to understand the BPM better. However, if these patterns are torn apart or distorted by bad layout decisions this advantage is lost (Schrepfer e.a., 2009).

Characteristics of the notation have a direct impact on comprehension of the model (Hahn & Kim, 1999). This leads to the conclusion that it is important to use the right notation for the right purpose.

Within the literature a lot of different influence factors are identified with regards to BPM understandability. These different views are tricky to combine because they either overlap or are so different. In the following paragraphs we attempt to make a clear overview of these different factors. The distinction is made between *model aspects* and *personal aspects*, very much like the divergence made by Reijers and Mendling (2011).

2.4.1 Model aspects

On the one side different model aspects influence BPM understandability, on the other side personal aspects have their effect as well. The model aspects that were identified in the literature are further discussed in this paragraph.

Lay-out

For example in the BPMN specification, there are no specifications made with regards to positioning of elements within the model. Since there are however specifications for the logic, two very different looking BPM (layout-wise) can actually withhold the same logic and elements and thus mean the same thing. Cognitive research in program understandability makes the distinction between the first notation and second notation of a program (Schrepfer e.a., 2009). The first notation of a modeling language can be seen as the specifications that are predefined in the language standards. The second notation is anything that the designer can decide for himself upon designing the BPM that are outside of the specifications of the standard. For BPMN there are no specifications with regards to positioning of the elements or in what direction the flow of the process should go. It has been proposed in research of process model understandability that the second notation can be expressed as the *amount of line crossings*, *edge bends*, *symmetry* and *use of locality* in the process model, however these aspects have not been empirically confirmed yet (Schrepfer e.a., 2009). Closely related to the second notation is the importance of flow direction in BPM (Figl & Strembeck, 2014). The authors propose a theoretical perspective on why a left-to-right direction is preferable for a BPM to a different direction, however the empirical evidence still needs to follow.

Granularity

A characteristic of BPM that distinguishes them from other forms of process modelling is that the description of activities are written in natural language (Gruhn & Laue, 2006a). This would imply that

not every activity has the same level of cognitive complexity. Gruhn and Laue make the assumption however that there should be an agreed-upon vocabulary that is used to express certain activities. Using some kind of glossary could increase the understandability of the BPM.

Modularity

Business process models can have specific parts of that can be regarded as a sub-process. Applying modularity to a BPM simply means dividing the large BPM into smaller chunks that can be seen as a subpart of the process. This way of designing should not only increase the comprehensibility of the model, but it should also make it easier to reuse parts of the model and make it more scalable (H. Reijers & Mendling, 2008). The advantages of the design mentality of modularity have already been demonstrated by the rapid growth of the computer industry (Carliss, Baldwin, & Clark, 1997). The findings of Reijers and Mendling (2008) in controlled experiments resulted to the conclusion that there seems to be a positive connection between the presence of modularity and the understandability of a process model. The effect seemed to be the most significant for larger BPM and if the design mentality is applied to a high extent. They found that the use of modularization techniques specifically increased comprehension of specific parts (locality) of the model. This modularization can be done in part automatically by software, and research results indicate that the use of sub processes increases the understandability of complex BPM due to the fact that they "hide" information from the user (H. A. Reijers, Mendling, & Dijkman, 2011).

2.4.2 Personal aspects

Not only the model aspects have their effect on the understandability of BPM. Since the understandability itself occurs at the user of the BPM, it would be nonsensical not to address any personal aspects. A popular personal aspect in the literature with regards to BPM understanding is expertise. Expertise can be regarded from two directions in BPM. From the one side, if the designer of BPM has a higher level of expertise he will probably produce higher quality models. A study by Bandara (2007) points out that if the user has more expertise in reading BPM or designing them, he will more easily understand complex BPM and thus make less errors when designing them. From the other side, if the user has more experience working with BPM he will understand them better. Defining expertise in the context of BPM seems to be a difficult task (Schrepfer e.a., 2009). Expertise not only consists of experience, but also theoretical knowledge (Mendling & Strembeck, 2008; H. A. Reijers & Mendling, 2011). In research around expertise the comparison between novices and experts is often made. However, this distinction does not offer much granularity when it comes to determining

the level of expertise. Experience with one graphical syntax or notation does not necessarily mean that the user will understand models in other notations (Nordbotten & Crosby, 1999). Nordbotten and Crosby (1999) refer to this as a carry-over effect between experience in different graphical syntaxes. Their research also indicated that more experienced users of a notation differ in reading strategy from their novice counterparts. However, this reading strategy did not seem more effective nor efficient for graphic-heavy models.

2.5 Conclusion

In the literature review we have uncovered the importance of BPM understandability and the relationship between BPM complexity and understandability is illustrated. This relationship has been the target of many studies in the field of process modelling and there seems to be a consensus about the positive correlation between the complexity of BPM and their level of understandability. Theories from cognitive sciences have been a popular reference point for the studies that attempt to address this relationship: Cognitive Load Theory, the SEQUAL model, semiotic theory, Resource Allocation Theory and the Cognitive Theory of Multimedia Learning have been reviewed on the level of their interpretation by researchers and what their conclusions imply for the understandability of business process models in general.

There are different factors that might influence the understandability of BPM according to the literature. In this review, the factors identified have been categorized as being either personal- or model-specific aspects. Model aspects that can influence the understandability are, amongst others, lay-out, granularity and modularity. A personal factor that has been identified by research is expertise,

Chapter 3: Measuring complexity

Understandability of a BPM is very closely linked to its complexity, as has been shown in the previous chapter. Therefore it is important to measure the complexity of a model in order to make sound estimations of its understandability. This paragraph gives an introduction into metrics that have so far been applied for measuring complexity of BPM, or process models in general. Most of these metrics are heavily inspired by equivalents used in software engineering. The preference is given to metrics using 'high-level' information of the BPM, because they make abstraction of the modeling language used. This increases the scope of their applicability which makes them more interesting for application in practice. However, these metrics often focus on just a few of the elements that actually influence the complexity of a BPM and therefore it is hard to pinpoint the single best metric. Using a combination of these metrics seems to be the preferred way to go (Gruhn & Laue, 2006b).

The different metrics that were identified from an extensive literature review are discussed as follows. Firstly, the metrics are discussed with regards to their origins and applicability to BPM. Secondly, for each metric a demonstration is given for the computational aspects using a fictive BPM example, that of a loan approval process. The example (Fig. 1) might seem arbitrary, but it is designed in such a way that it allows for demonstration of the computational aspects for each metric. It contains three subprocesses, being the Home Loan (Fig. 2), Student Loan (Fig. 3) and Car Loan Approval (Fig. 4) processes. Because of the limited space in a printed document we have opted for contracted subtasks in the main process model and the subprocesses are displayed separately in their full form. For the computation of the various metrics we however assume that the process model is displayed in its full form as it can be found in the Appendix on page 74.

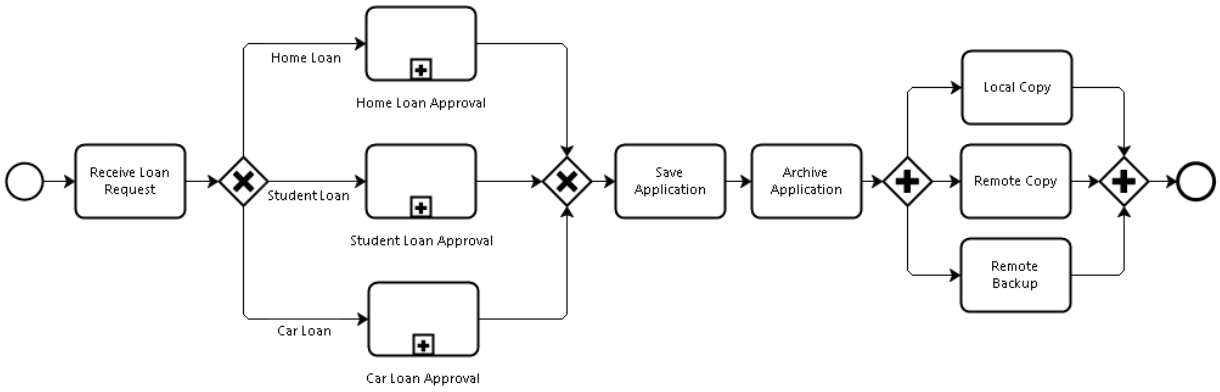


Fig. 1: Loan approval BPM example

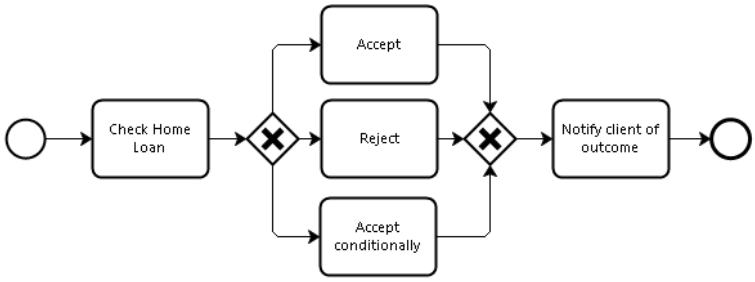


Fig. 2: Home Loan subprocess

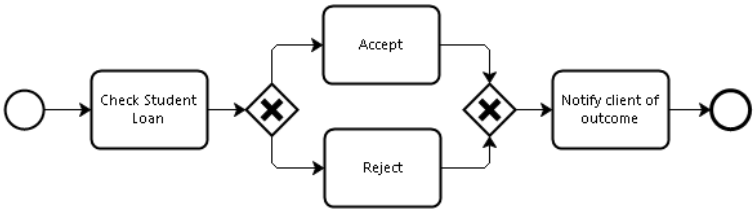


Fig. 3: Student Loan subprocess

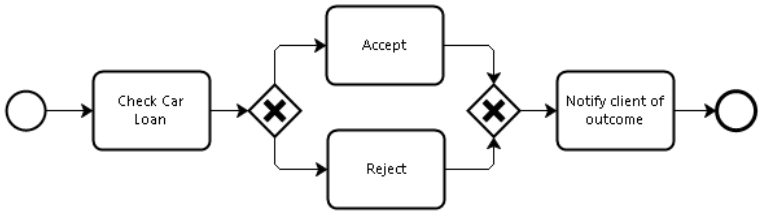


Fig. 4: Car Loan subprocess

3.1 Lines of Code (NOA, NOAC, NOAJS)

The Lines of Code (LOC) metric is probably the easiest of all metrics discussed in this chapter. However, its simplicity does not impede its usefulness. It is a basic metric well known from its applicability in software engineering for detecting errors. The name of the metric says it all: the amount of lines of the code gives an indication of the complexity of the software. This metric is easily translated in a form in which it is applicable for BPM. One possible definition for the LOC metric is given by Cardoso (2005). According to him, the elements of a BPM can be assigned to these subsets: Number Of Activities (NOA), Number Of Activities And Control-flows (NOAC) and lastly Number Of Activities, Joins and Splits (NOAJS). The metric can also be linked to Cognitive Load Theory, as discussed earlier, as the number of elements in a BPM create a cognitive load on the working memory of the user. It is the long-term memory (expertise) that enables the user to recognize patterns and this way reduce the cognitive load on his working memory to free up resources for understanding the model. Which in a way is linked to Resource Allocation Theory (Mendling e.a., 2012). The LOC metric might be a decent metric for complexity, however, it does not offer that much depth as a measure of *cognitive* complexity. In a BPM there are many more factors (logic, lay-out, notation, …)

that increase the cognitive complexity and far better metrics have been established by research in this subject. Some examples are discussed further on.

3.1.1 Computational aspects

Since there are no lines of code in a graphical representation of a BPM, we resolve to the metrics that were adapted from LOC for BPM by Cardoso: number of activities (NOA), number of activities and control flows (NOAC) and number of activities, joins and splits (NOAJS).

The NOA can be easily computed for the loan request process as can be found in Fig. 1 on page 17. To find the value for the NOA we simply count the number of tasks, which are the equivalent form of activities, in the BPM. With the calculations for this metric the assumption is made that an 'activity' refers to the task element in the BPMN notation. We make this assumption because of the simple reason that if it would refer to both tasks and gateways, the NOA would be equal to the NOAJS. This assumption, however, is not made for all the other metrics. For this reason we explain for each metric individually what is meant with the abstract concept of 'activities'. The main process consists of 9 tasks and the subprocesses consist of respectively 5, 4 and 4 tasks. This results in a value of 22 for the NOA metric.

The computation of NOAC is very similar to the NOA metric. We can recycle the 22 value of the NOA metric and simply add the total amount of control flows in the BPM. The control flow amount is computed by simply counting the amount of arrows in the BPM found in Fig. 1. The main process contains 18 control flow arrows. The subprocesses contain respectively 10, 8 and 8 control flow arrows. This leads to a total of 44 control flow arrows. When we add this to the 22 tasks (NOA) we end up with a value of 66 for the NOAC.

For the calculation of the NOAJS we yet again recycle the value for the NOA metric (22). We then add the number of joins and splits. In BPMN these joins and splits are represented by gateway joins and splits. The main process has two XOR-gateways and two AND-gateways, both splits and joins. The subprocesses have two XOR-gateways each (again split and corresponding join). This brings the total splits and joins of the BPM to 10. The value for the NOAJS can be seen as the sum of the amount of tasks and gateways in the BPM, which is then 32.

3.2 Control-flow Complexity (CFC)

Based on McCabe's Cyclomatic complexity (McCabe, 1976), a popular metric from software engineering, the Control-flow Complexity metric measures the complexity of process models

(Cardoso, 2005). This metric is based on the amount of XOR/OR/AND-splits and joins in the process model and it counts the total states that are possible due to these splits and joins. The amount of states of the model is equal the different paths or flows that are possible in the model. XOR-splits represent an activity that only allows the control flow to go in one of many possible directions. The XOR-join is an activity that can have multiple incoming directions, but is started as soon as one token arrives at the join. The AND-split only starts execution when all incoming connections are completed and then starts all outgoing connections. The OR-split can enable any amount of outgoing flows and can be enabled from any amount of incoming flows.

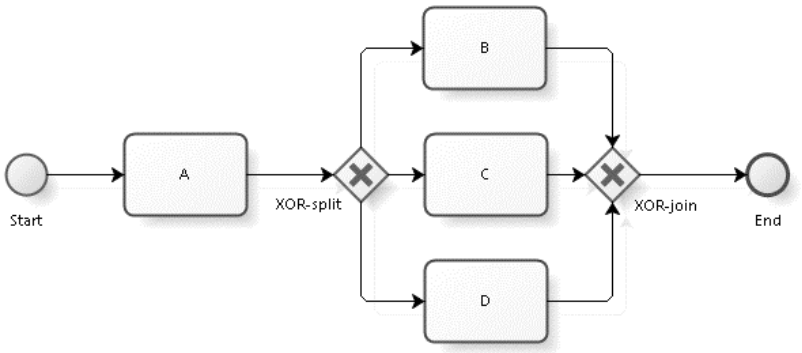


Fig. 5: XOR-split and XOR-join in BPMN 2.0

As can be seen in Fig. 5 the XOR-split is represented in BPMN and many other process modelling languages as a gateway with an X in it. The logic of the XOR should have additional information what single path should be chosen. This means that only the activity B or C or D can be executed when the process comes to an end. The XOR-join continues the flow as soon as one incoming flow is registered, meaning if either B, C or D is completed the flow will continue, and in this case will reach the end of the process.

The way the XOR-split's complexity is measured correspondents with the number of fan-outs of the split. With 'fan-out' is meant the amount of flows go from the XOR-split to other activities. Cardoso (2005) sees these fan-outs as the number of states that the designer of the BPM needs to keep in mind and analyze. These states are the number of different possibilities or paths that the process model allows for.

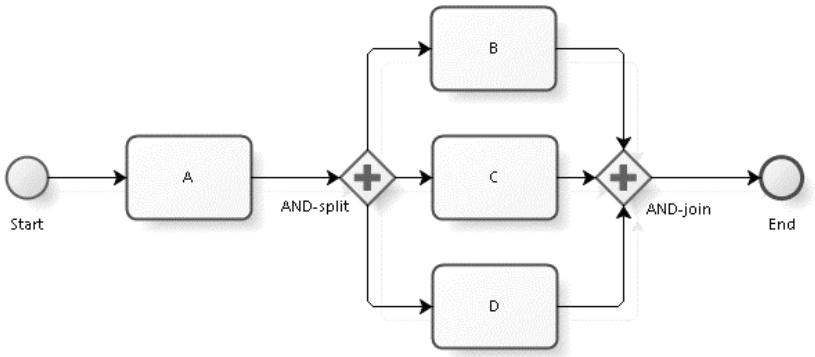


Fig. 6: AND-split and AND-join in BPMN 2.0

As demonstrated in Fig. 6, the AND-split and join are illustrated in BPMN 2.0 as a gateway with a plus-sign in it. The logic of the AND-gateway dictates that all outgoing flows are started as soon as the gateway is reached. This means that activities B, C and D will be executed in parallel. The AND-join will only activate (and continue the flow) when the activities B, C and D are finished.

The complexity of an AND-split is easily measured as being equal to 1. This is due to the fact that the AND-split does not increase the number of states, since all the fan-outs (flows) from the AND-split are followed and no other possibilities are allowed by the model.

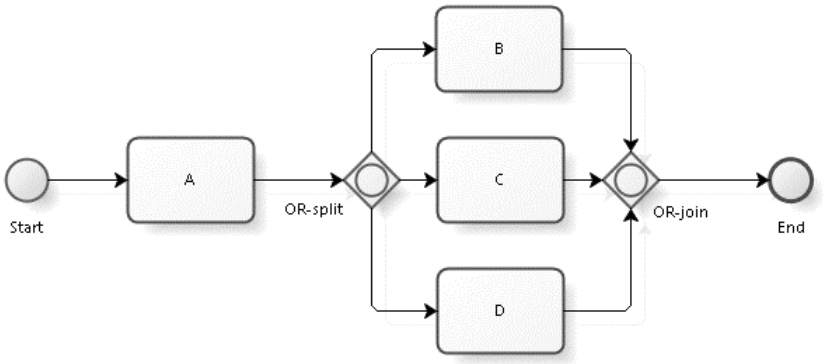


Fig. 7: OR-split and OR-join in BPMN 2.0

The OR-split in BPMN 2.0 looks like a gateway with an O inside it. The OR-split works differently than the AND or the XOR-split. The OR-split works in such a way that it needs to be defined in the logic of the activity which of the outgoing flows should be continued. In this case it can be one, two or three different flows. The same goes for the OR-join, the flow will continue when the amount of incoming flows needed is reached.

The OR-split is probably the most complex of the different types of splitting in process models according to Cardoso. The complexity of the split is equal to 2^f-1 where f equals the amount of fan-outs that start from the split. The reason for this is because there are more paths possible within the model due to the use of the OR-split.

Cardoso has validated the metric by applying Weyuker's formal list of properties (Weyuker, 1988). These properties were constructed to give an analytical approach to validate software metrics for their quality. The metric satisfied seven out of a total nine properties and therefore it can be considered as valid. Furthermore it has been confirmed by an experiment that resulted in the conclusion that the metric is very highly correlated with the user's perceived complexity. This makes the metric also an interesting one with regards to the understandability of BPM. There is a disadvantage to the metric however according to the author. It lies in the fact that there is no real objective meaning to the actual CFC-value of a process model. To give meaning to the value it needs to be a subject of empirical studies or real-world experience to give classifications of (for example) risk associated with specific levels of CFC. This would be very likely as what has been done around the McCabe complexity metrics: certain values for the metric would indicate higher amounts of complexity and risk associated with the software program.

The metric is also further criticized by Gruhn and Laue (2006b): the metric does not account for the structural complexity of the model. Two models with the same CFC-value but different structures or lay-outs are demonstrated and there seems to be an obvious difference in the level of cognitive complexity, even though the CFC-values are completely the same. Although the CFC-metric can be useful to gain insight of the complexity with regards to logic and maybe give an idea of the cognitive complexity, it should be used with caution to measure the overall cognitive complexity of a BPM.

3.2.1 Computational aspects

The CFC metric as proposed by Cardoso is a little less straightforward than his NOA, NOAC and NOAJS metrics. For the XOR-gateways we need to count the total amount of *fan-outs* Within BPM, these fan-outs are the number of control flows leaving the exclusive choice gateway. The example as given in Fig. 1 has one XOR-gateway split with a corresponding join in the main process. Since there are three different control flows leaving the gateway, we should assign the value of 3 to the CFC metric for this XOR-gateway. Next up (in the main process) is an AND-gateway. According to the metric's specification, the AND-gateway represents a CFC of 1. This results in a CFC value for the main process of 4. The subprocesses have one XOR-gateway split and join each. However, the fan-outs are not the same. The first subprocess has a gateway with three control flows leaving, which means a value of 3 should be added to the total CFC. The other two subprocesses have gateways with two control flows leaving, which means that a value of 2 should be added for both subprocesses. This results in

a total CFC of 11 for the complete BPM example. This calculation is based however on the assumption that the total CFC of the BPM is the sum of the main process and its subprocesses complexity.

3.3 Cognitive Weights (CW)

As a result from empirical studies, Shao and Wang (2003) defined a metric to measure the comprehensibility of a piece of software. They do this by assigning varying weights to different control structures. These weights are ordered by difficulty of understanding, as demonstrated in Table 1. The total cognitive weight of a software component is then defined by the sum of the structure weights, assuming there are no nested control structures.

Table 1: Cognitive Weights, weights according to control structure

Structure	*Weight (W_i)*
Sequence	1
Branch with if-then or if-then-else	2
Embedded function call	2
Branch with case	3
Iterations	3
Embedded recursion	3
Concurrency	4

The cognitive weights metric can be applied to BPM, however some considerations should be made in order to adapt the metric (Gruhn & Laue, 2006b). For example, recursion has no meaning in the context of BPM. There are also aspects inherent of BPM that are not represented by a cognitive weight. An example of this is a cancellation within a BPM. The adoption by Gruhn and Laue (2006a) of the Cognitive Weights for BPM can be found in Table 2.

Table 2: Cognitive Weights adapted by Gruhn and Laue (2006a) for BPM

BPM Control Structure	*Weight (W_i)*
Sequence	1
XOR-split, one of two chosen, with join	2
XOR-split, one of ≥ 3 chosen, with join	3
AND-split with join	4
OR-split with join	7

Subtask	2
Multiple Instance Activity	6
Cancellation	1, 2 or 3

Gruhn and Laue (2006a) point out however that the Cognitive Weight metric should be used with caution when the BPM is unstructured. A structured BPM is one in which each gateway split (AND, XOR, …) is matched with the corresponding join gateway and that the split-join pairs are properly nested (Liu & Kumar, 2005). Various studies however have pointed out that unstructured BPM could be translated into well-structured BPM most of the time (Kiepuszewski, ter Hofstede, & Bussler, 2000; Liu & Kumar, 2005). This could increase the usefulness of CW as a metric to assess BPM complexity and thus understandability.

3.3.1 Computational aspects

Calculating the value for CW of a BPM is pretty straightforward since the specifications made by Gruhn and Laue (2006a) are very clear. For specific parts of the BPM a weight is assigned. The sum of these weights results in the value of CW for the BPM, as can be seen in Table 2 above. We apply this to the example as given in Fig. 1. A *sequence* (W_i = 1) can be seen as a sequence of tasks in BPMN. We have one sequence in the main process model and none in the subprocesses. The main process has one *XOR-split-gateway with one of ≥ 3 chosen*, which results in a W_i of 3. It also has an *AND-split with join* which accounts for a W_i of 4. There are three *subtasks* in this model which results in a W_i of 6. There are no Multiple Instance Activities, Cancellations or OR-gateways in this BPM so we do not have to add weights for these types of structures. The subprocesses however do contain XOR-gateways. The first subprocess (Fig. 2) has a XOR-gateway with one of three possible control flows, which results in a W_i of 3. The other two subprocesses have a XOR-gateway with one of two possible control flows, which results in a W_i of 2. This brings the grand total CW of the BPM to 21. Again, just as with the CFC metric, the assumption is made that the total CW of the model is the sum of the CW of the main process and its subprocesses.

3.4 Nesting depth (ND)

Following from (again) research in software development, it has been found that both the mean and maximum nesting depth have a strong correlation with structural complexity (Schroeder, 1984). This metric is easily adopted for BPM, since both metrics (mean nesting depth, maximum nesting depth)

can be easily defined (Gruhn & Laue, 2006b). The depth of an activity can be seen as the amount of decisions that have to be made to get to this activity in the BPM. Although it should be noted that some business modeling languages (UML AD, YAWL) do not need 'proper' nesting of activities (Gruhn & Laue, 2006b). With *proper nesting* the authors mean that not every split or join needs to occur in pairs. According to Gruhn and Laue this is comparable with GOTO-jumps in software programming languages as being unstructured, and WHILE and FOR-loops being structured loops. When a lot of these jumps occur it affects the structure of a BPM and the occurrence of a 'spaghetti'-model might take place (Holl & Valentin, 2004). Another metric that is proposed by Grun and Laue (2006b) is the knot count of a BPM. A knot occurs when the control paths intersect with each other, as demonstrated in Fig. 8.

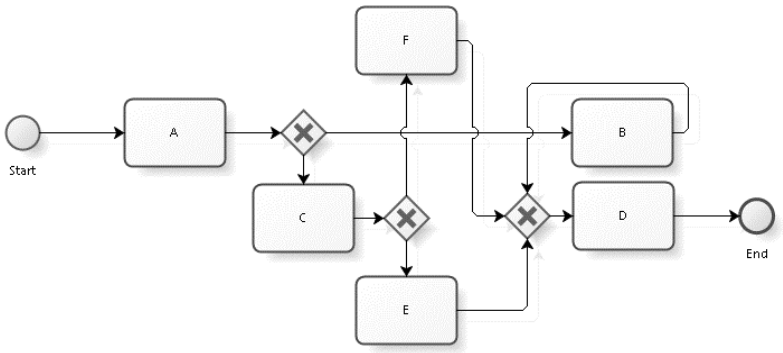


Fig. 8: Knots in a BPMN 2.0 model

The amount of knots in a BPM could have an impact on the understandability of the model, however in this domain (to my knowledge) no specific research has been done. One would expect that the relationship between the amount of knots and the understandability of a BPM is negative, however this assumption would need to be validated through empirical research.

3.4.1 Computational aspects

Firstly we compute the *mean nesting depth* (MeanND) of the example BPM as can be found in Fig. 1. We can do this without any problems because the example fits the *proper nesting criterion* as described above. We look at the tasks (which are here considered as 'activities') in the BPM and how far they are *nested* within the model. As explained earlier, the nesting depth is the amount of decisions that need to be made before the activity is reached. For the "receive loan request" task, no decisions need to be made for the activity to start. The other tasks in the main process model have a nesting depth of zero: no decisions need to be made for them to be reached, they are not nested. However, the nesting depth is different for the tasks in the subprocesses. Since the subprocess itself is nested within an XOR-gateway split and join, all the tasks within the subprocess

already have a floor of 1 nesting depth. For example, in the Home Loan Approval subprocess (Fig. 2), the "Check home loan" task is not nested in the subprocess, but since the subprocess itself is nested it attains a nesting depth value of 1. The "Accept", "Reject" and "Accept Conditionally" tasks however are yet again nested inside XOR-gateways. This means for example that in order to reach the "Accept" task, two specific decisions need to be made. This results in a nesting depth value of 2 for these tasks. The same can be said for the other subprocesses (Fig. 3 and Fig. 4). We then sum up these nesting depths and divide them by the total amount of tasks in the BPM. The total nesting depth for this BPM is then 20 and the total amount of tasks is 19. This results in a mean nesting depth of 1,05.

The maximum nesting depth was found for the tasks nested within the XOR-gateways in the subprocesses. This results in a value of 2 for the MaxND.

3.5 Information Flow (IF)

Based on earlier research by Henry and Kafura (1981) in software development, Gruhn and Laue (2006b) adapt the fan-in and fan-out metrics for application to BPM. Fan-in in terms of software systems means how many times a piece of the software is being addressed by other pieces of the same software. *Vice versa*, the fan-out is the amount of other modules of the software are being addressed from a specific module. This way both metrics give an idea about the structural complexity of the software since it addresses in which degree it is modularized (Henry & Kafura, 1981). The same can be said for a BPM: if a node has a lot of different incoming and outgoing flows, this increases the complexity of the BPM. If the structural complexity is high due to this metric, it most likely means that the designer of the model did a poor job (Gruhn & Laue, 2006b). Based on earlier work from Shepperd (1989), Sun & Hou (2014) define a new metric (Information Flow) for measuring complexity of process models. The individual IF of an activity is calculated by multiplying the fan-in with the fan-out then taking the square. This metric is very close to how Cardoso e.a. proposed the concept of Interface Complexity (IC), but it is much easier to compute than Cardoso's alternative.

3.5.1 Computational aspects

For the IF metric we regard both the tasks and gateways as activities, this in contrast to the NOA/NOAC/NOAJS and ND metrics. We make this assumption because we want to employ the same computation method as was used by the authors who proposed this metric (Sun & Hou, 2014). The calculation of the IF of sub-activities or sub-processes is the sum of the IF of all the activities

contained. To calculate the total IF all the individual elements' IF are summed up. When we look at the main process (Fig. 1) and the subprocesses (Fig. 2, Fig. 3, and Fig. 4), we can already see that all the tasks only have one control flow coming in and going out. This means that the flow is *controlled*. This results in an IF of 1 for all the tasks in the model. The first XOR-split-gateway has one incoming flow and three outgoing flows. This means that the IF for the first gateway is $(1+3)^2$ or 16. The same value is computed for the XOR-join-gateway. Up next we still have an AND-split and join-gateway, for which the same IF is computed (16). This results in a total IF for the main process of 50 (6 for the tasks, 16+16 for the XOR-gateways and 16+16 for the AND-gateways). This demonstrates that the metric does not distinguish between AND or XOR-gateways. This could be seen as a disadvantage of the metric. AND-gateways can introduce parallelism into the BPM and even multiple instances when they are not merged properly (synchronization). Intuitively one would expect that a BPM with either parallelism or even multiple instances within the same process flow would increase the cognitive complexity of the BPM. Since this metric does not account for this increased cognitive complexity it would seem straightforward that additional metrics are needed that do address this aspect of complexity. The subprocesses have a total of 13 tasks that have one incoming and outgoing flow. The first subprocess (Home Loan Approval) has a XOR-split with one incoming and three outgoing and a XOR-join with three incoming and one outgoing flow, which implicates an IF value of 32. The second and third subprocess have XOR-gateways with only two outgoing and incoming flows. This means that for each of the two subprocesses the gateways account for an IF value of 9. The grand total of the IF metric for this BPM is the sum of the main process and subprocess IF. This can be easily computed by adding 50 (main process), 13 (tasks subprocesses), 32 (XOR join-split first subprocess), 18 (XOR join-split second and third subprocess) which results in an IF value of 113.

3.6 One for all, or all for one?

As has been pointed out above, different metrics can address different aspects of the complexity of BPM. However, with regards to total cognitive complexity of a BPM it makes little sense to apply only a single metric. An interesting point of view by combining metrics is presented using the Goal-Question-Metric (Abd Ghani, Koh, Muketha, & Wong, 2008). GQM is a paradigm based on defining project goals and then constructing questions that need to be asked in order to achieve said goals. The next step is then to fit the questions with metrics such as those that have been mentioned in this chapter. This approach is comparable to that of a Balanced scorecard (BSC): in a BSC various

financial and non-financial metrics are gathered in a small report for management to assess the current status of operations. Other interesting combinations have been proposed in the literature, such as combining Cardoso's Control-flow Complexity with Nesting Depths (H. A. Reijers e.a., 2011). While the CFC-metric expresses the overall logical complexity, the ND-metric measures the structural complexity. Gruhn and Laue (2006a) also expressed their concerns about using the metrics individually in assessing complexity of a BPM. They too suggest that the individual metrics should complement each other and should be used within a metrics suite to gain insight of the overall cognitive complexity of a BPM.

For a summarization of all the complexity metrics discussed above, see Table 3.

Table 3: Summary of complexity metrics for (business) process models

Metrics	*Authors*	*Usage*
NOA, NOAC, NOAJS	*Cardoso (2005)*	NOA: Count number of activities/tasks NOAC: Count tasks and control flows NOAJS: Count activities and joins and splits
CFC	*Cardoso (2005)*	Total states XOR/OR/AND-splits/joins
CW	*Shao and Wang (2003)*	Adding weights to elements of the model
ND	*Gruhn & Laue (2006b)*	Mean nesting depth, maximum nesting depth, knot count
IF	*Sun & Hou (2014)*	Information flow, product of individual Fi and Fo squared.

3.7 Conclusion

Since the complexity and understandability of BPM seem to be so closely related, some of the most popular metrics for process model complexity are summarized and reviewed. These metrics, due to the close relationship between complexity and understandability, could give sound estimations of the understandability of BPM. Some of the metrics seem intuitively more suitable for measuring BPM complexity than others and sometimes even combinations might be preferable. Combining the metrics can be done for example through the Goal-Question-Metric methodology, which is comparable to the popular Balanced Scorecard method. This leads to the question whether some of these metrics actually measure the same thing, which would make their concurrent use nonsensical. However, the combination of some of these metrics might actually make sense. This leads us to the

next chapter, where the metrics are evaluated on the basis of Workflow Patterns (Van der Aalst) expressed in BPMN 2.0.

Chapter 4: Patterns

4.1 Introduction

To make the link with the previous chapter, the different levels of understanding (syntax, semantics and pragmatics) are discussed in this chapter according to specific patterns that have been identified in the literature. The use and reuse of patterns can help make the BPM more understandable because it allows the user to divide the model into sizeable "chunks" (Cardoso e.a., 2006). The metrics from Chapter 3 are applied as a basis for evaluation of their cognitive complexity. The results of this comparison should give insights into the understandability of BPM and can be regarded as an evaluation on the usefulness of metrics. Three popular streams for patterns were identified in the literature, namely *Workflow Patterns*, *Activity Patterns* and *Domain Process Patterns*. These groups of patterns can be linked to the levels in the theoretical ladder of Burton-Jones e.a. (2009) as discussed in Chapter 2: Theories on understandability. According to the authors, understanding has to go through different levels: syntax, semantics and pragmatics. The three groups of patterns that were identified can be linked correspondingly to these levels and will be individually discussed below.

Workflow Patterns are often-recurring patterns that address business requirements in a workflow-like expression but not in any specific notational language (W. M. P. van der Aalst e.a., 2003). The reason the authors refrained from expressing the patterns in a specific notation is because they wanted to create a basis for comparing modeling languages and workflow management systems. The 20 different patterns described in their paper (W. M. P. van der Aalst e.a., 2003) range from rather simple to complex. This collection of workflow patterns has been expanded and revised (N. Russell, 2006). The total of 43 identified patterns have been documented and their ability to be expressed in various notations was tested by Russel (2006). These patterns provide insights at the syntax-level of the BPM. Use and re-use of these patterns might increase the understandability of BPM since the users will recognize them and this might reduce the cognitive load with regards to syntax (Cardoso e.a., 2006)

While Van der Aalst's (2003) Workflow Patterns focus on the syntactical level of process models, *Activity Patterns* (AP) are aimed at the semantic level of BPM (Thom, Reichert, & Iochpe, 2009). The authors address semantics because they identify key business activities from frequently recurring BPM-patterns. The goal of their research is to encourage re-use of BPM patterns which would allow for the design of higher quality BPM. In their paper they bring a revised version of the seven activity

patterns. The distinction between Van der Aalst's WFP and Thom, Reichert and Iochpe's AP can be easily made. While WFP are primarily aimed at the most fundamental syntactical structures in a BPM, AP add more meaning to often recurring patterns of activities in BPM. Examples of such patterns can be approval processes, notification processes, question-answer processes… By the use of semantics (*send notification, deny request, …)* meaning is added to the patterns. This places AP higher on the ladder in semiotic theory of understandability (as discussed in Chapter 2: Theories on understandability) than WFP. This is logical, because WFP are far more abstract in their definition.

in their paper "Improving the process of process modelling by the use of *Domain Process Patterns (DPP)*", Koschmider and Reijers (2015) introduce process patterns for specific business domains. These process patterns differ from Van Der Aalst's Workflow Patterns (WFP) in such a way that WFP focus at the syntactical level, as described in the previous chapter, while the DPP focus at the content of the process. At the semantics level, Thom e.a. (2009) proposed Activity Patterns. With these patterns typical business functions were associated. For example, according to the authors a decision could be identified by a XOR-split. With DPP the focus is at the activity level in a modeling domain. This means that the context is important for the pattern, bringing it to the pragmatic level. The goal of DPP is to assist designers in the creation of BPM by supplying predefined content in the form of reusable process patterns. The domains of focus are order management and manufacturing, however the collection of DPP is obviously extendable to other domains. This is what distinguishes DPP from AP: DPP adds context to recurring patterns, allowing it to reach a higher level on the semiotic ladder: the pragmatics level.

Due to the constraints on time and resources associated with the master's thesis, the scope will be limited to the original set of Workflow Patterns (20) by Van Der Aalst (2003). The link with the literature review can be made with the various complexity metrics that were discussed and how they influence the syntax level of understanding. Since the complexity metrics discussed in the previous chapter are closely related to the syntax of process models, it makes sense to apply these metrics to patterns that make syntax the focal point. The WFP contain frequently returning arrangements found in BPM. These patterns range from very simple, to very complex, and therefore are an interesting subject of study in comparing different cognitive metrics for process model understandability. Future research could address higher levels of understanding: semantics and pragmatics. Correspondingly, Activity Patterns and Domain Process Patterns could be an interesting starting point for these higher levels of understanding.

4.1.1 Methodology

Firstly, the different patterns are described and a fictive business situation is given to explain their functionality. Secondly, the various metrics discussed in Chapter 2 will be applied to the different patterns. For the notation of these patterns, where possible, BPMN 2.0 is used[3]. Because the patterns are not explicitly expressed in a language, they need to be 'translated' to BPMN in order to apply the metrics. Some of the process diagrams are inspired by or transcribed from Steven White's paper on BPMN-diagrams (2008). A table is then constructed summarizing the results of this analysis. Firstly, a conclusion is drawn from the results of these metrics. Secondly, the patterns are discussed with their applicability to the syntax level of understandability and how their usage can improve the understandability of BPM. It could be that some patterns have higher-than-necessary complexity and should therefore be avoided or replaced in the design phase of BPM to increase understandability. Thirdly, results of the metrics applied to the different patterns are compared based on their correlation coefficient and a factor analysis is conducted. Higher correlations between metrics could indicate that they measure the same aspect of BPM complexity, therefore concurrent usage of the metrics would be nonsensical and could or should be avoided.

The following metrics are applied:

Abbreviation	*Metric*
NOA	Number of Activities
NOAC	Number of Activities and Control Flows
NOAJS	Number of Activities and Joins and Splits
CFC	Control-Flow Complexity
CW	Cognitive Weight
MeanND	Mean Nesting Depth
MaxND	Max Nesting Depth
IF	Information Flow

[3] For the BPMN 2.0 specifications the reader is referred to the website of the Object Management Group (http://www.omg.org).

4.2 Workflow Patterns

The Workflow Patterns (WFP) originally identified by Van Der Aalst have been divided under six categories: *basic control flow, advanced branching and synchronization, structural, multiple instance, state-based,* and lastly *cancellation patterns*. The different patterns range from very simple to very complex and they try to cover popular returning arrangements in business process models. Every pattern is briefly explained and the pattern's translation into BPMN is given in a figure. Then a table is shown with the value for each metric that was computed. For the computational aspects for each metric the reader is referred to Chapter 3: Measuring complexity.

4.2.1 Basic Control Flow

Sequence

The *Sequence* pattern is by far the simplest of all the workflow patterns. It consists of two or more activities that are executed consecutively. A practical business process example could be that the activity *Send invoice* is executed after *Ship order*. These activities are done in sequence and they are connected with arrows in BPMN. In Fig. 9 an example is shown. Activity B cannot start until A has ended, and C will start when B has ended.

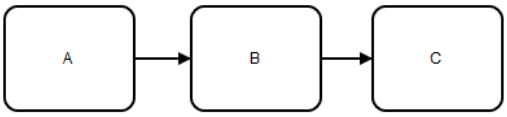


Fig. 9: Sequence Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Sequence	3	5	3	1	1	0	0	3

Parallel Split

The *Parallel Split* pattern allows the process to have multiple activities executed simultaneously. In comparison with the previous example, when the customer's order is shipped (*Ship order*) not only should the invoice be sent (*Send Invoice*), but at the same time the customer should be informed of the shipment (*Inform Customer*). Sending the invoice and informing of the shipment should happen in parallel and thus a control flow pattern is needed to express this relationship in BPM. There are three ways to design such a pattern in BPMN, as can be seen in Fig. 10, Fig. 11 and Fig. 12. The first example is simply forking the control flow from A to B and C. The user has to know what the BPMN guidelines dictate with such a fork in order to understand the parallel relationship of B and C. The

second option in Fig. 11 is seen as a 'best practice' (White, 2008). The AND-gateway dictates that the control flow is split up in two parallel threads that can be executed simultaneously. These multiple threads can be *synchronized* later on with the Synchronization pattern. The third option (Fig. 12) is perhaps a bit far-fetched, but still valid according to the BPMN specification. It makes use of a sub-process to start two new threads and supports the same functionality as the options in Fig. 10 and Fig. 11. The downturn of using this construction is however, that the parent-process will not continue until all threads in the sub-process have ended and this could introduce errors in the design phase of the BPM.

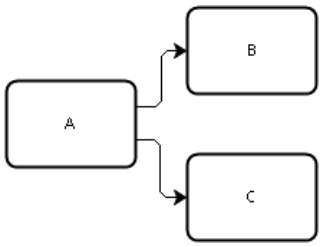


Fig. 10: Parallel Split Pattern (option 1)

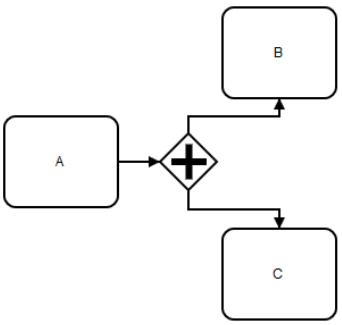


Fig. 11: Parallel Split Pattern (option 2)

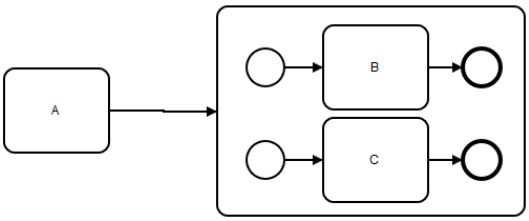


Fig. 12: Parallel Split Pattern (option 3)

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Parallel Split 1	3	5	4	1	2	0	0	6
Parallel Split 2	3	6	4	1	2	0	0	7
Parallel Split 3	3	8	4	1	4	0	0	3

Synchronization

The *Synchronization* pattern is the exact opposite of the Parallel Split. It joins two or more process threads back into one control flow. In Fig. 13 and Fig. 14 examples of such a pattern in BPMN are shown. In the first example an AND-gateway is used to merge the control flows arriving from activity A and B. Both A and B need to be completed before C will be executed. In the second example a sub-process is used. When the sub-process is activated activities A and B are started (see *Parallel*

Split), when both are ended then the control flow from the sub-process can continue to start activity C. An example of such a pattern in a business context could be that the order cannot be shipped (Ship order) before the goods are packaged (Package goods) and the shipping bill is made (Create shipping bill), which can happen simultaneously.

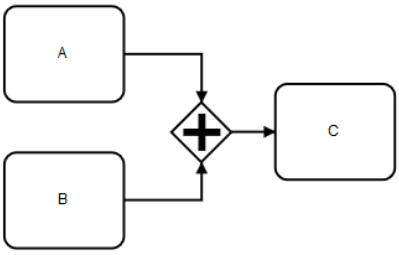


Fig. 13: Synchronization Pattern (option 1)

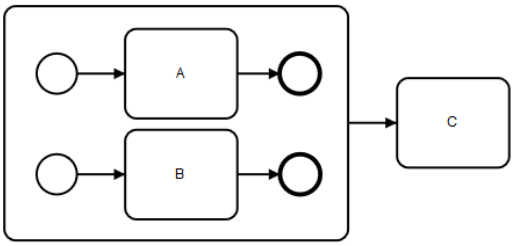


Fig. 14: Synchronization Pattern (option 2)

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Synchronization 1	3	6	4	1	2	0	0	7
Synchronization 2	3	8	4	1	4	0	0	3

Exclusive Choice

The *Exclusive Choice* pattern expresses a decision in a process model. A decision has to be made between several different branches. In BPMN this is done by the usage of an XOR-gateway, as demonstrated in Fig. 15. If condition X is met, activity B is executed. If condition Y is met, then C will be executed. Both conditions have to be mutually exclusive, otherwise it is not an Exclusive Choice pattern, but a *Multi-Choice* pattern, which will be discussed later on. This pattern is used when decisions need to be made based on *a priori knowledge*. This means that the designer knows how the decisions will be made when he is designing the BPM. This decision is however not based on business logic, but more so on process logic. A business process situation where such a pattern would appear is for example when a customer has the choice to either pick up the order himself or that he wants his order to be shipped to him. If the customer wants to pick up the order himself, a different path in the process needs to be followed than when he wants it to be shipped. Other activities need to be undertaken in both situations and the customer cannot combine both options: an *exclusive choice* needs to be made.

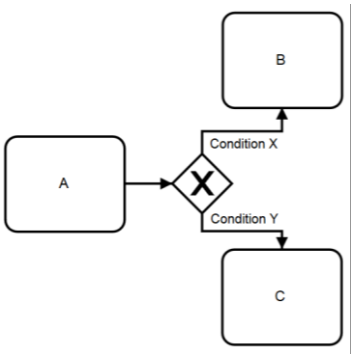


Fig. 15: Exclusive Choice Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Exclusive Choice	3	6	4	2	1	0,66	1	7

Simple Merge

The *Simple Merge* pattern is a bit more complex. This is because the pattern allows for multiple instances to continue, this in contrast with the *Synchronization* pattern. Examples of such a pattern in BPMN are shown in Fig. 16 and Fig. 17. As the first example shows, it is possible to do this by using regular flow arrows from activity A and B to C. However, to someone not completely familiar with the BPMN specifications, it is not clear whether A and B need to be completed before C can start. According to the BPMN specifications this kind of notation would imply that C can start whenever A or B is completed. This would mean that multiple threads of the process can exist: C can be executed multiple times if A and B both complete. This is referred to as an *uncontrolled* flow. The second option with the XOR-gateway is much more straightforward. If the user understands the logic of the XOR-gateway he should understand that the completion of A or B would start activity C. In a business process it would make sense to let the *Exclusive Choice* pattern be followed by the *Simple Merge* pattern. To continue from the example given with the Exclusive Choice pattern: whether the customer picked up the order himself or it got shipped to him, an invoice still needs to be sent to him. However, these cases are all under the assumption that only one token arrives at activity C. When this is not the case and the model designer does not anticipate for this, there might be problems. If somehow both paths would be followed in the business case, it would result in the customer receiving two invoices and the shipment being sent to him at home.

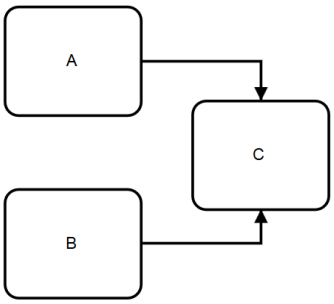


Fig. 16: Simple Merge Pattern (option 1)

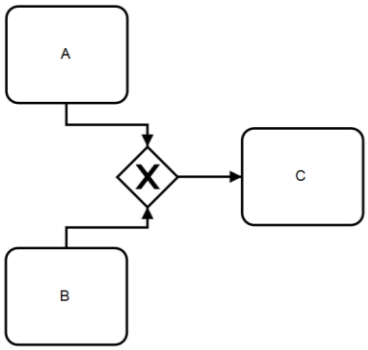


Fig. 17: Simple Merge Pattern (option 2)

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Simple Merge 1	3	5	4	2	1	0	0	6
Simple Merge 2	3	6	4	2	1	0	0	7

4.2.2 Advanced Branching and Synchronization

Multi-Choice

The *Multi-Choice* pattern occurs when one or more paths can be followed based on conditions in the BPM. An example of this functionality within BPMN is given in Fig. 18. In BPMN this type of control flow is demonstrated by usage of an OR-gateway. In the example given below, activity B will be executed when condition X is met. If condition Y is met, activity C will be executed. The conditions do not have to be mutually exclusive, this in contrast with the *Exclusive Choice* pattern: both conditions can be met and this way the process can be split up into multiple threads from the point of the gateway. A business process situation where such a pattern occurs can easily be demonstrated. For example the customer could order different types of products that need very different treatment with regards to packaging, shipping, … He could have chosen products from different types, or just one specific type, still the necessary activities based on the types of products he ordered need to be executed.

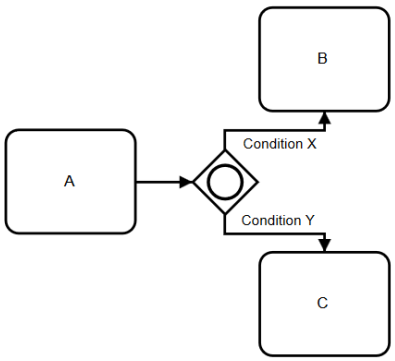


Fig. 18: Multi-Choice Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Multi-Choice	3	6	4	3	3,5	0,67	1	7

Synchronizing Merge

The *Synchronizing Merge* is the more complex counterpart of the Multi-Choice pattern. The modeling of the pattern in BPMN as it was defined by Van Der Aalst (2003) is not evident. An example is given in Fig. 19. The OR-gateway is used with supplemental logic added inside. The logic states that the gateway should wait till all 'needed' threads have completed. The 'needed' threads are those that have been started after the split occurred. This pattern is also based on the assumption that every thread is only completed once. This means that a limitation is put on the model that the threads can only complete once while the gateway is waiting or the other threads to finish. While the graphical BPMN model seems very simple, the logic that is within the OR-gateway is not. It might be hard for users to understand. For the business process example the situation is easier to sketch. Continuing from the previous example at the Multi-Choice pattern, the customer could have ordered different types of products that have different shipping requirements (for example product-specific packaging). If the customer ordered only one type of product, then the OR-gateway will continue the flow if that particular thread is completed. If the customer ordered more different types, the OR-gateway will wait for all the active threads to complete before continuing the flow.

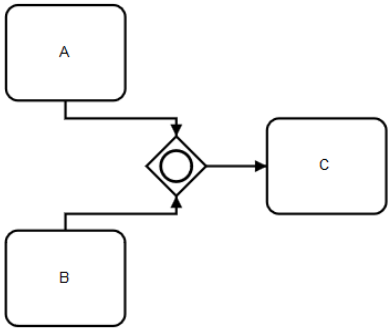


Fig. 19: Synchronizing Merge Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Synchronizing Merge	3	6	4	3	3,5	0,33	1	7

Multi-Merge

Another example of the previous mentioned *uncontrolled* flow, is the *Multi-merge* pattern. This pattern occurs when two activities or two threads converge again, but without the synchronizing effect. This means, for example in the BPMN model as can be seen in Fig. 20, that activity C can be executed more than once. Just like with the first example of the Simple Merge, this behavior might not be completely clear to someone who is not familiar with the BPMN specifications, or it could cause confusion to those who are familiar with it. It is considered 'best practice' to use an XOR-gateway (White, 2008), like in Fig. 17. While the Simple Merge has different assumptions about the preceding activities and threads, the pattern looks the same in BPMN as it does with the Multi-Merge (Wohed, van der Aalst, Dumas, ter Hofstede, & Russell, 2005). With regards to understandability the same proposition is made by White (2004): it might be confusing for users to employ such patterns, as it allows for multiple executions of the same activity in the workflow. An example with regards to a business process could be that some parts of the packaging process are the same for two different types of products. This would mean that they need to undergo the same activity, but in a different instance.

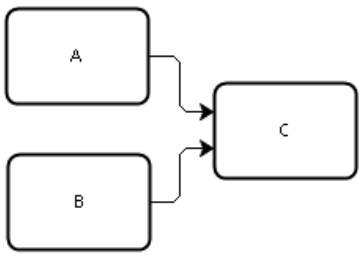


Fig. 20: Multi-Merge Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Multi-Merge	3	5	4	2	8	0	0	6

Discriminator

The Discriminator pattern expresses a functionality in the process that should continue the flow as soon as one incoming thread has completed. All future incoming threads will be ignored. This means that the total concurrent threads will be reduced to one. In some way, it is comparable to the Synchronization pattern, however only one preceding activity needs to be completed. There are different ways to express this functionality in BPMN. The first possibility is as has been described by White (2004), with the usage of an XOR-merge gateway with a label added that the nature of the gateway is in fact discriminating. However, this method has been criticized by Wohed e.a. (2005). In their paper they regard the Discriminator pattern in BPMN as a special case of the N-out-of-M Join pattern. The N-out-of-M Join pattern is a pattern that "depicts the ability of synchronizing a flow after N parallel threads (out of M initiated threads) have completed" (Wohed e.a., 2005). In normal language, this means that the flow of activities will only continue when all the needed preceding activities have been completed. The amount of needed preceding activities (N) is not necessarily equal to the total possible routes of activities that could have been taken earlier (M). The Discriminator pattern is comparable with an N-out-of-M Join pattern in such a way that N equals exactly to 1. This means that all the other paths that have been taken will be ignored by the gateway and that only the first thread is registered and continues the flow of activities and all other threads that arrive later will be discarded. This interpretation of the pattern in BPMN can be found in Fig. 22. The authors propose another solution through the use of an empty activity after an XOR-merge gateway. For example in a business process context, this pattern could present itself in a situation where multiple departments (in a BPM divided into lanes) could have the rights to make decisions. It could be possible that one department would make a decision sooner and the request is approved and the rest of the process can continue. However, for such a situation it might be better to make use of Cancellation patterns that cancel the activities in other departments with regards to the decision making. These types of patterns will be discussed later on.

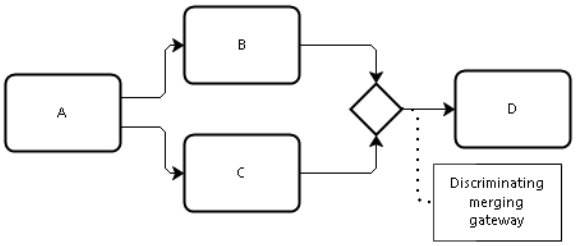


Fig. 21: Discriminator Pattern according to White (2004) (1)

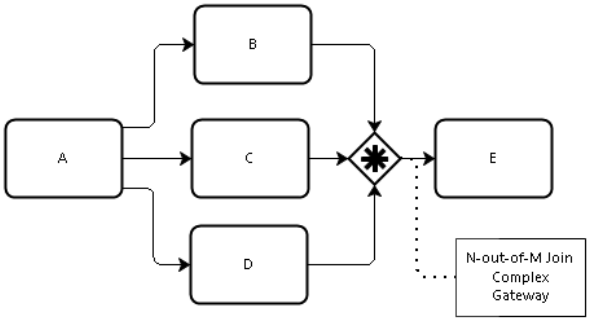


Fig. 22: Discriminator Pattern according to Wohed e.a.(2005) (2)

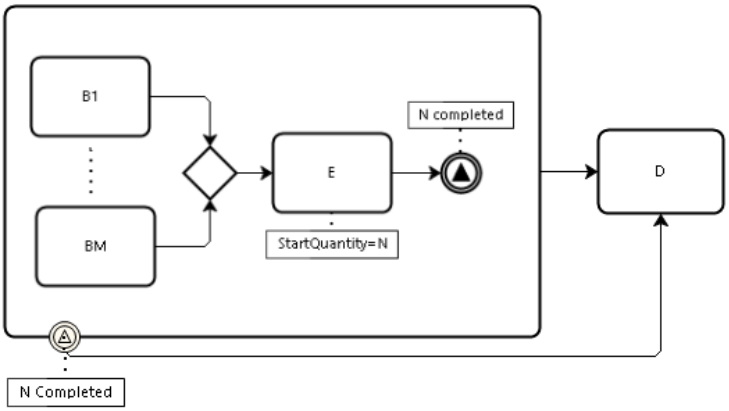


Fig. 23: Discriminator Pattern according to Wohed e.a. (2005) (3)

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Discriminator 1	4	9	6	3	3	0,25	1	11
Discriminator 2	5	12	7	4	5,5	0,2	1	22
Discriminator 3	4	10	5	3	3	0,5	2	10

4.2.3 Structural

Arbitrary Cycles

The *Arbitrary Cycles* pattern expresses a point in the BPM where some activities (not all) can be done multiple times. An example of such a pattern can be seen in Fig. 24. In this example, when activity C is finished the decision is made whether to 'go back' to activity B or continue to activity D. Activity B is an *upstream* activity (White, 2008). An important aspect of this pattern is that the loop itself has an alternative exit node, in this example this is after activity F has been completed. A business process example could be a situation where quality checking is done of products in a manufacturing environment. Activity B could be a quality check. If the quality is sufficient, the flow will go to activity F *(Add to stock*). If the quality is not sufficient, activity C should happen (*Fix quality problem*). If the

item is fixed, it should go back to the quality check (B). If the item is not fixed, it should go to D (*Production failure handling*).

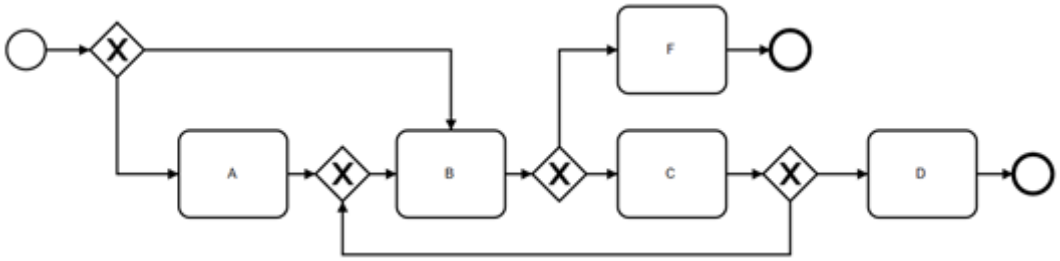


Fig. 24: Arbitrary Cycles Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Arbitrary Cycles	5	13	9	6	4	2	3	24

Implicit Termination

The Implicit Termination pattern expresses the functionality of a BPM to end the entire process when there are no other (parallel) activities needing completion. BPMN completely supports this kind of logic because BPMN supports different kinds of end-nodes. The None-end-node ends the current thread, however the Terminate-end-node ends all threads. In this pattern the assumption is made that there are no other active threads in the BPM, thus normal (None) end-nodes are sufficient. An example of such a pattern in BPMN is given in Fig. 25. A business situation for this kind of pattern is easily sketched: if there is nothing left to be done, the process ends.

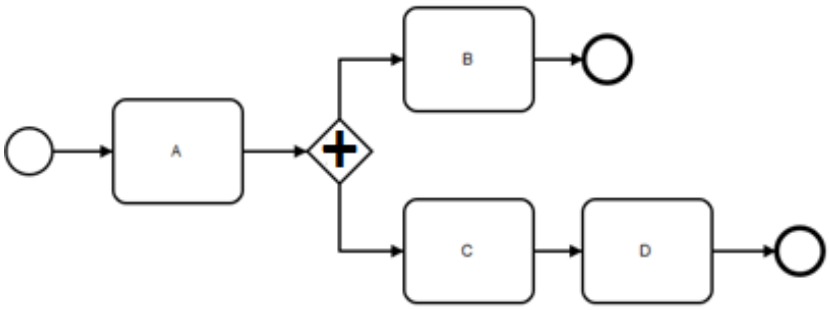


Fig. 25: Implicit Termination Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Implicit Termination	4	11	5	1	3	0,75	1	8

4.2.4 Multiple Instances (MI)

With *a priori* Design-Time Knowledge

This pattern is the kind where the designer of the BPM already knows how many times a specific activity needs to be repeated in the business process. This means that before (*a priori*) the model is designed (*design-time*) the amount of repetitions is known (*Knowledge*) to the designer of the BPM. In a business context such a pattern would present itself when an activity needs to be repeated a known number of times before continuing to the next activity. An example in a manufacturing environment could be product testing. The designer of the BPM knows how many the product needs to be tested by the employee and inputs it into the model. An example of such a pattern can be seen in Fig. 26.

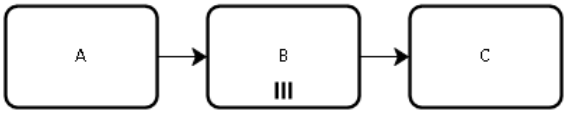


Fig. 26: Multiple Instance Pattern with a priori Design-Time Knowledge

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
MI with a priori Design-Time Knowledge	3	5	3	1	8	0	0	3

With *a priori* Run-Time Knowledge

This pattern is different from the previous pattern in such a way that the designer does not know how many times the activity should be repeated when he is designing the BPM. The amount of repetitions is dependent on a condition that should be specified before (*a priori*) the activity's execution (*Run-Time*).

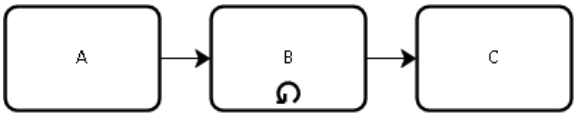


Fig. 27: Multiple Instance Pattern with a priori Run-Time Knowledge

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
MI with a priori Run-Time Knowledge	3	5	3	1	8	0	0	3

Without *a priori* Knowledge

This patterns is again different from the two preceding patterns. Within this pattern, the amount of repetitions of a certain activity is not known before the activity is undertaken. This functionality is a bit more complex to construct in BPMN, but an example of such a pattern is given in Fig. 28. In the example, two parallel activities (B and C) are started. Activity C is the activity where the amount of iterations of the other activity (B) is determined. This means that the use of an XOR-gateway is needed after activity B with two conditions. If the determined amount of iterations by C for B is reached, the flow will continue to D. If there are not enough copies of B, the thread will end after B. However, the token coming from activity C will continue to another XOR-gateway with two conditions that follows the same logic as the other XOR-gateway: if enough copies of B are present the flow will continue to activity D. If this is not the case, an *upstream* loop is made. A business context that would require such functionality in a BPM could be similar to the example given for the *a priori Design-Time knowledge* pattern. Testing products could depend on the manufacturing data. If a lot of deviations were detected in the manufacturing process, most likely there will be more defects. If the risk for defects is higher, maybe the amounts of iterations of the testing phase should be increased to avoid faulty productions.

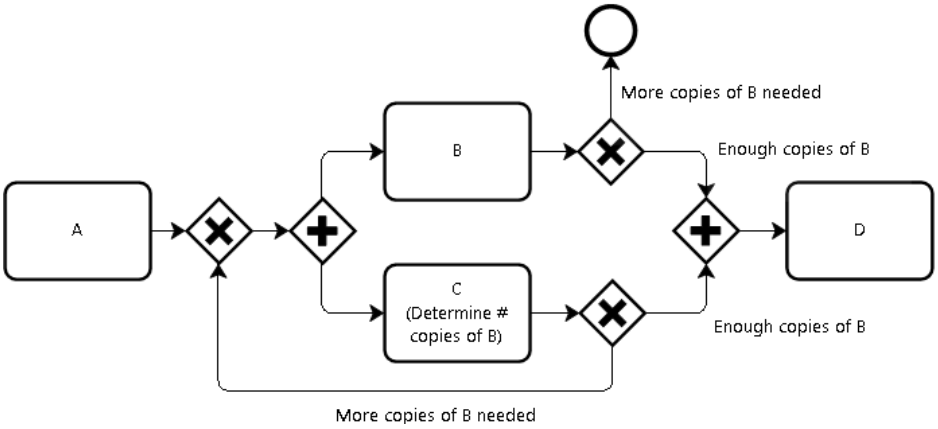


Fig. 28: Multiple Instance Pattern without a priori Knowledge

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
MI without a priori Knowledge	5	15	9	6	6	0,5	2	23

Requiring Synchronization

In the first two MI patterns discussed the need for synchronization was addressed. If this need is present, this should be added into the BPM. To address this in the BPMN notation, extra information

is needed for the parallel looping logic of the activity. This can be done by setting the "*MI_FlowCondition*"-variable to "*ALL*". In the example shown in Fig. 29, this would mean that all iterations of B need to be completed before activity C can start.

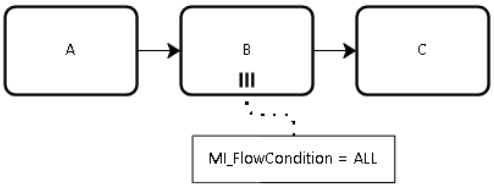


Fig. 29: Multiple Instance Pattern Requiring Synchronization

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Requiring Synchronization	3	5	3	1	8	0	0	3

4.2.5 State-based

Deferred Choice

The *Deferred Choice* pattern is quite similar to the Exclusive Choice pattern. However, the difference is in the fact that the decision is made in a different way. With the use of the Exclusive Choice pattern, the decision is based on process data. The Deferred Choice pattern in contrast is based on the occurrence of events. This means that if the event occurs, the path is taken and the other possible control flow paths are disregarded. In BPMN this can be easily modeled using an Exclusive Event-based gateway, as can be seen in the example in Fig. 30. The gateway after activity A will wait for either 24 hours passing by, which would result in activity B commencing, or for a confirmation, which would result in activity C in executing. It is straightforward to sketch a business situation where such a pattern would occur. It could be that a sales rep made an offer to a customer for a particular order of goods. If the customer does not reply within a timeframe of 24 hours, the offer expires (B). If he does reply, other activities (C) need to be undertaken.

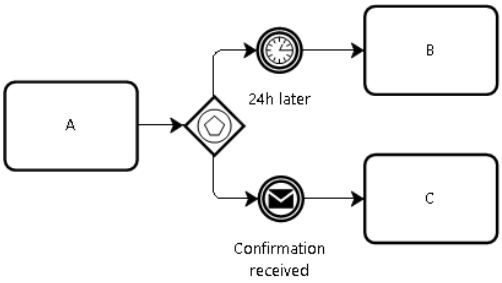


Fig. 30: Deferred Choice Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Deferred Choice	3	8	4	2	3,5	0,67	1	9

Interleaved Parallel Routing

The *Interleaved Parallel Routing* pattern occurs when two or more activities can be done consecutively, however their order is irrelevant and is decided on upon run-time. An example in BPMN is given by White in Fig. 31. The notation for such a pattern can be realized by the usage of an ad-hoc sub-process. The sub-process needs to be defined further in such a way that the activities should be executed sequentially (and which). A business context example could be the process for hiring new staff. It could be that new staff need to undergo a personality and analytical test. They cannot be done at the same time by the same person, but the order is irrelevant. The process cannot continue unless both tests have been completed.

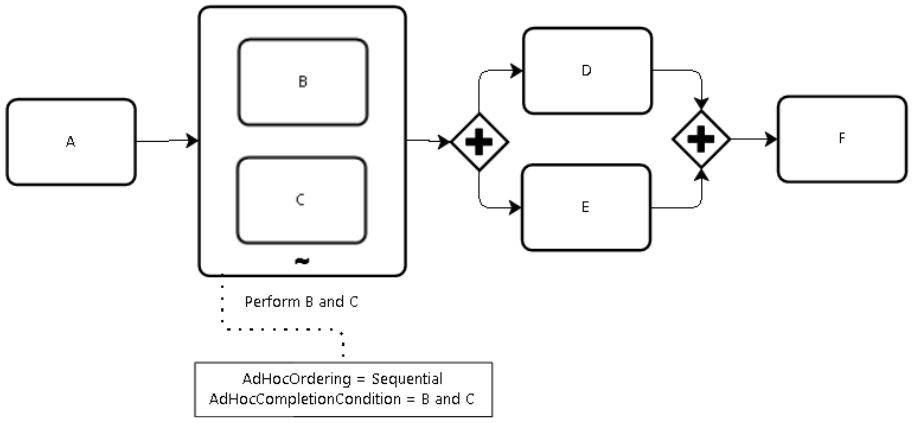


Fig. 31: Interleaved Parallel Routing Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Interleaved Parallel Routing	6	13	9	3	6	0,17	1	16

Milestone

The *Milestone* pattern occurs when the end of a particular activity in the process implies other activities in parallel to be activated, or to be made impossible to reach after its expiration. An example by White (2004) is presented in BPMN in Fig. 32. This example by White is however criticized by Wohed e.a. (2005). According to the authors White's example does not allow for expiration of the milestone, a crucial aspect of the pattern. They propose a different solution which is reprinted in Fig.

33. They circumvent the inability of BPMN to express states of activities by using a messaging mechanism between sub-processes to inform about the state of completion on activity B. An example within a business context could be that the customer can cancel his order until 48 hours before the delivery date. If he does not cancel his order within this time period, the delivery will be made. The milestone here is the starting of the 48 hour-period before the delivery date. Once this milestone is reached, the customer cannot cancel his order anymore.

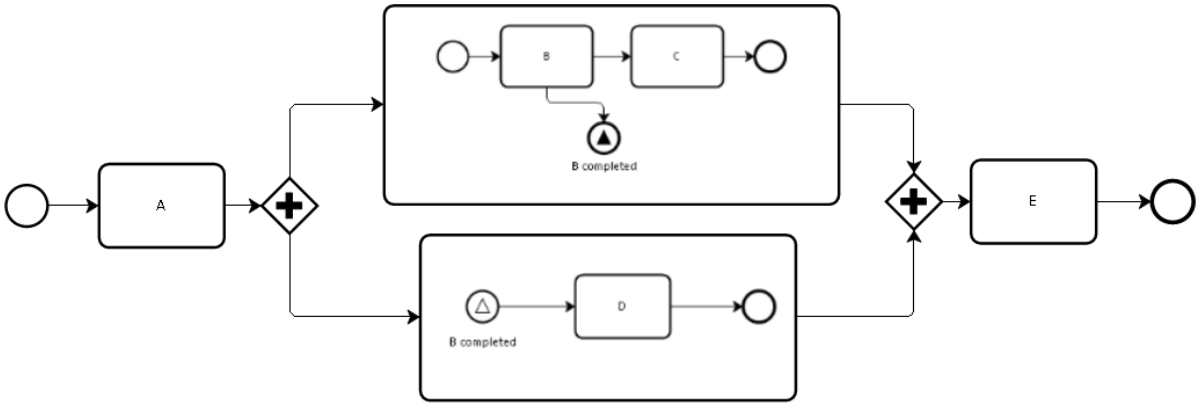


Fig. 32: Milestone Pattern by White (2004) (1)

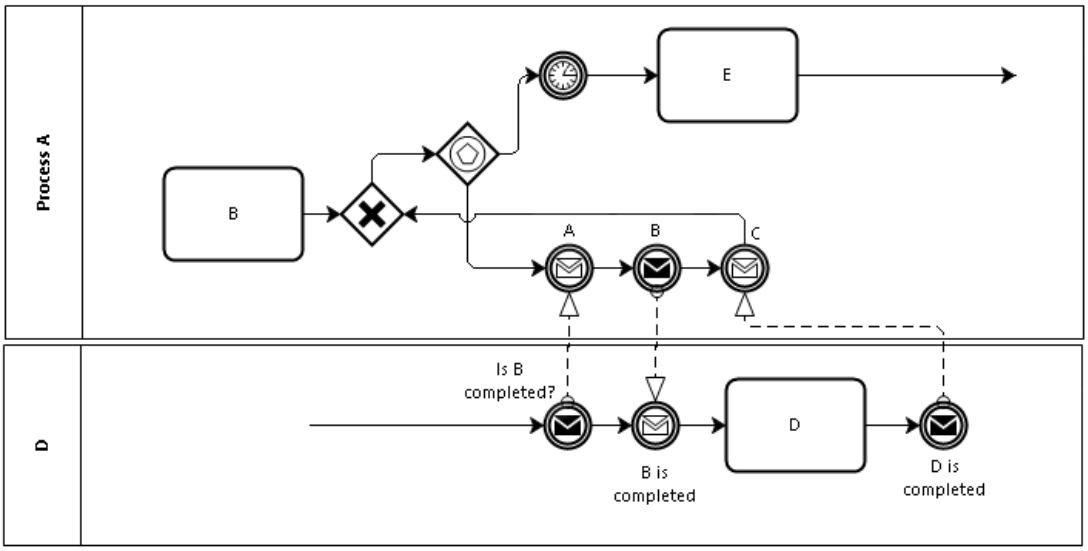


Fig. 33: Milestone Pattern by Wohed e.a. (2005) (2)

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Milestone 1	5	19	8	2	11	0,17	1	16
Milestone 2	3	16	5	3	3,5	0,33	1	14

4.2.6 Cancellation

Cancel Activity and Cancel Case

The *Cancel Activity* pattern occurs when there are two (or more) competing activities. When one of the two is completed, the other should be interrupted. In BPMN this can be done by using an interrupting intermediate event at the second activity, which is activated by a signaling event after the first activity. An example of such functionality can be seen in Fig. 34. If the first activity (B) is finished, the second activity (C) is then canceled. A business situation where such a pattern might occur could be when there are two parallel activities and one activity is completed, the other should be stopped. The *Cancel Case* pattern is almost identical to the Cancel Activity pattern. The only difference is that the activity C is replaced by a sub-process C.

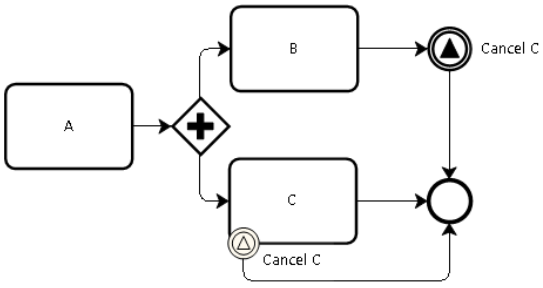


Fig. 34: Cancel Activity Pattern

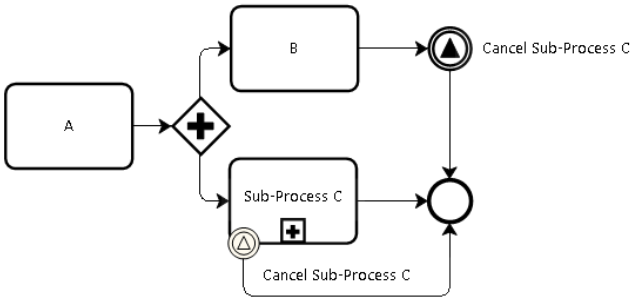


Fig. 35: Cancel Case Pattern

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
Cancel Activity	3	10	5	2	4,5	0	0	9
Cancel Case	3	10	5	2	6,5	0	0	9

4.3 Analysis

4.3.1 Descriptive statistics

First of all some descriptive statistics are generated of the data that have been collected for the different Workflow Patterns. These descriptive statistics can be found in Table 4. The number of activities (NOA) does not differ much between the different patterns. This is easily explained, as most patterns do not contain that many activities because the focal point of the patterns are on the control flow which in BPMN is constructed by gateways and control flow arrows. This is also the reason why the number of activities and control flows (NOAC) does vary in greater amounts between the different

patterns. The highest NOAC was measured for the Milestone 1 pattern. The second highest score was assigned to the Milestone 2 pattern. If the NOAC alone could be considered as a good means of measuring complexity, the Milestone patterns could be regarded as the most complex patterns discussed here. For the number of activities, joins and splits (NOAJS) the same can be said as with regards to the NOA: there is not much variation between the different patterns. The Arbitrary Cycles pattern scores the highest NOAJS, which could indicate that it contains a lot of gateway-logic. The Control-flow Complexity (CFC) is the highest (6) for this pattern as well. The Cognitive Weight metric assigns the highest value to the Milestone 1 pattern as well. This could confirm the original indication by the NOAC-metric. However, the danger exists that these two metrics actually measure the same thing, we try to test this later by looking at the Pearson correlation coefficients between the different metrics and by applying factor analysis to the data. The mean and maximum nesting depth (MeanND/MaxND) of the patterns do not vary a lot, however these metrics have deeper meaning and any difference should indicate a greater difference in complexity. The assumption for now is that not all the metrics are equally sensitive, which can be easily demonstrated by comparing the IF and ND values. This assumption is however still preliminary and further analysis needs to be conducted in the next paragraphs. The highest nesting depth (3) was realized in the Arbitrary Cycles pattern, which at the same time was assigned the highest mean nesting depth. The Information Flow (IF) metric is the metric that has the highest variation. The reason for this can be found in the way it is calculated: increased number of control flows have an exponential influence on the metric in absolute value.

	N	*Minimum*	*Maximum*	*Mean*	*Std. Deviation*
NOA	27	3	6	3,52	0,89
NOAC	27	5	19	8,63	3,91
NOAJS	27	3	9	4,93	1,86
CFC	27	1	6	2,22	1,40
CW	27	1	11	4,31	2,62
MeanND	27	0	2	0,27	0,43
MaxND	27	0	3	0,63	0,79
IF	27	3	24	9,22	6,15

Table 4: Descriptive statistics of the measurements

Because the different metrics measure comparable aspects of complexity, they can be analyzed based on their correlations. If some metrics demonstrate high correlation coefficients, this might indicate that the different metrics measure the same aspect of complexity, or that the different aspects measured are linked together. In Table 5 the Pearson correlation coefficients can be found. The single and double asterisks indicate significance at the 5% and 1% level respectively. From this

table we can see that the NOA, NOAC and NOAJS demonstrate high correlations, which was expected as all three of them account (partly) for the amount of activities in the patterns. Anything below a 0,6 correlation value can be considered as low or moderate correlation.

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
NOA	1							
NOAC	,728**	1						
NOAJS	,927**	,826**	1					
CFC	,644**	,565**	,777**	1				
CW	,321	,358	,269	,059	1			
MeanND	,400*	,348	,481*	,657**	,133 (-)	1		
MaxND	,663**	,599**	,712**	,843**	,003	,862**	1	
IF	,825**	,794**	,923**	,876**	,179	,564**	,775**	1

Table 5: Pearson Correlation Matrix

Since all of the metrics discussed here attempt to capture the same concept (complexity) of BPM, their low correlation might indicate that they measure different aspects of the same concept. To find out what these aspects are we conduct a factor analysis in the next paragraph.

4.3.2 Factor analysis

Because the results from the descriptive statistics confirmed our initial expectations (the metrics measure different aspects of complexity), we conduct a factor analysis. A factor analysis can be used to identify the dimensions of complexity that are measured by the metrics. This is done by employing the *principal components* method. This method tries to identify whether some variables belong to a specific dimension by identifying linearly uncorrelated variables. The Kaiser stopping criterion is employed to decide on the amount of dimensions to be extracted: all factors with eigenvalues greater than 1 should be ignored. The scree plot generated from SPSS can be found in Fig. 36.

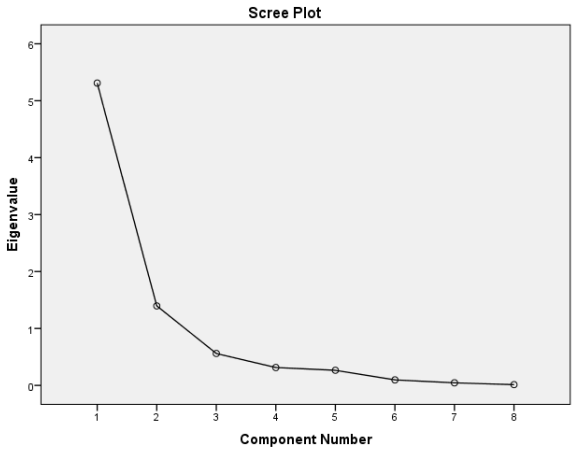


Fig. 36: Scree Plot from Factor Analysis

The scree plot demonstrates an almost ideal pattern: it starts with a steep decline, then bends into a somewhat straight line. As can be interpreted from the scree plot, the flat straight line starts from the third component. This means that the metrics we have applied to the different patterns can be reduced to two different dimensions.

Total Variance Explained

Component	Initial Eigenvalues			Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	5,310	66,376	66,376	4,393	54,918	54,918
2	1,396	17,451	83,828	2,313	28,910	**83,828**
3	,561	7,006	90,834			
4	,314	3,923	94,757			
5	,265	3,311	98,068			
6	,096	1,205	99,273			
7	,045	,564	99,837			
8	,013	,163	100,000			

Extraction Method: Principal Component Analysis.

Table 6: Results from Factor Analysis: Total Variance Explained

When we look at the total variance explained by these two components (dimensions) of complexity, we find that these two groups account for a cumulative total of 83,828% variance explained, which is rather high. This means that these two components of complexity that were identified account for 83,83% of the variation found in all studied complexity metrics.

Rotated Component Matrix[a]

	Component	
	1	2
NOA	,635	**,654**
NOAC	,552	**,689**
NOAJS	**,735**	,623
CFC	**,880**	,232
CW	-,202	**,826**
MeanND	**,874**	-,171
MaxND	**,946**	,118
IF	**,820**	,494

Extraction Method: Principal Component Analysis.

Rotation Method: Varimax with Kaiser Normalization.

a. Rotation converged in 3 iterations.

Table 7: Results from Factor Analysis: Rotated Component Matrix

From the Rotated Component Matrix (Table 7) we can deduct that some of the metrics belong to the first component group and other metrics belong to the second component group. This means that

the metrics from different groups most likely measure different aspects of complexity, but that the metrics that belong to the same group measure most likely the same aspect. This answers one of our research questions: some metrics actually seem to measure the same thing based on the factor analysis. However, the factor analysis also pointed out that the metrics that were tested most likely measure two different dimensions of complexity. NOAJS, CFC, MeanND, MaxND and IF are placed in the first component group. These metrics are often associated with structure and control flows. NOA, NOAC and CW are placed in the second component group and are not as much linked to flows but rather to the nature of the elements within the BPM. This could mean that the first group could be collectively seen as a measure of *structural complexity*, while the second group could represent measures for *functional complexity*.

To answer the question *whether the different metrics assign the same ranking to the patterns*, we look at the Spearman Ranked Correlation Matrix as can be found in Table 8.

	NOA	*NOAC*	*NOAJS*	*CFC*	*CW*	*MeanND*	*MaxND*	*IF*
NOA	1							
NOAC	,715**	1						
NOAJS	,819**	,903**	1					
CFC	,578**	,597**	,720**	1				
CW	,252	,230	,166	,092	1			
MeanND	,529**	,556**	,523**	,646**	,097 (-)	1		
MaxND	,721**	,681**	,685**	,782**	,020	,936**	1	
IF	,757**	,862**	,924**	,801**	,102	,646**	,779**	1

Table 8: Spearman Ranked Correlation Matrix

Overall, the ranking of the patterns by the different metrics seem to be relatively highly correlated at the 1% significance level. This would mean that the different metrics assign somewhat the same ranking to the different patterns, since they are highly (positively) correlated. This answers our research question: we can conclude that the different metrics assign highly correlated rankings to the different patterns, which means that they should assign rankings in a similar way. An obviously high ranking correlation is between the NOA, NOAC and NOAJS: they all account for the amount of tasks/activities in the BPM. CFC is highly correlated with NOAJS. This was to be expected as well, because both of them measure structural complexity of the BPM. Less straightforward is the high (positive) correlation between the MeanND and NOA. One explanation could be that the more activities are present within the BPM, the more decisions are needed to get to these activities. More interesting relationships can be found with regards to the IF metric. It is highly correlated with all

the metrics, except for CW. Even though not significant, it does offer interesting perspectives for further research.

4.3.3 Pattern comparison

To compare the different patterns with regards to their measured complexity we transformed the data into different forms. These forms should provide a straightforward basis for comparing the patterns. Firstly, a line graph comparison is made based on unweighted values (Fig. 37). These values come straight from the computed metrics for each pattern. They are sorted based on their weighted values, because this creates a more logical order in the patterns going from rather complex to simple. The weighted values were computed by weighing the metric values measured per pattern to the metric's maximum found in the descriptive analysis. Because the weighted values comparison shows different results (graphically) from the unweighted values, we can conclude that the metrics are not equally sensitive. This already answers our research question (*are all metrics equally sensitive?*) negatively. For example, as can be seen in Fig. 37, the IF metric is very sensitive, which was to be expected because of its quadratic nature. Because of these differences in sensitivity a comparison is made based on the weighted values of the metrics, as can be found in Fig. 38. Still, this graphical comparison does not distinctively show the differences between the complexity metrics for the different patterns: a general trend is still rather hard to identify.

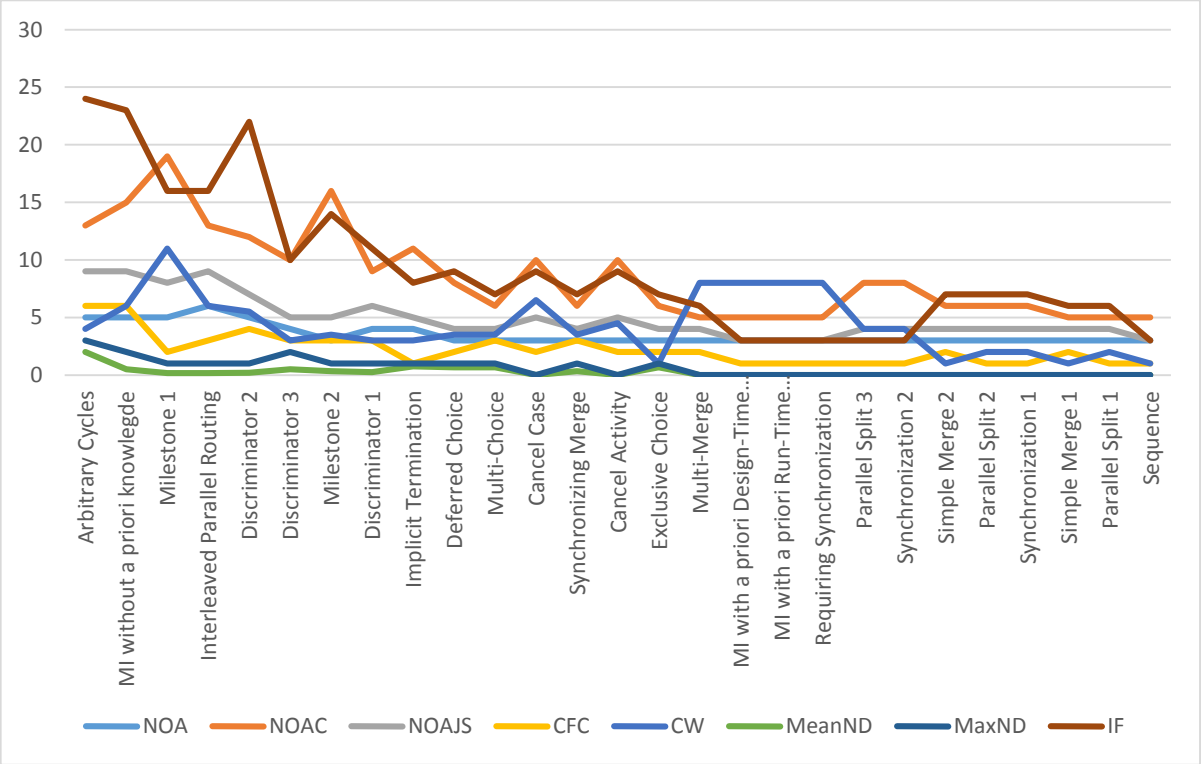


Fig. 37: Pattern comparison (unweighted values)

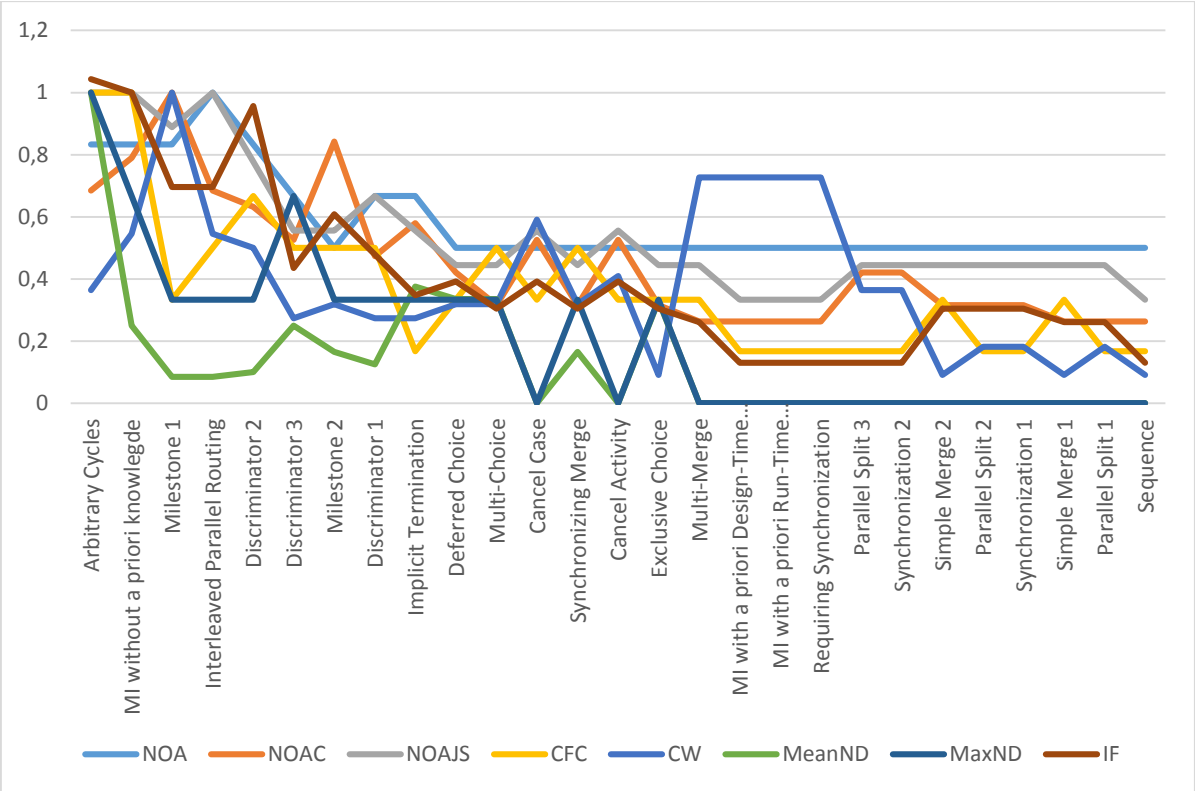


Fig. 38: Pattern comparison (weighted values)

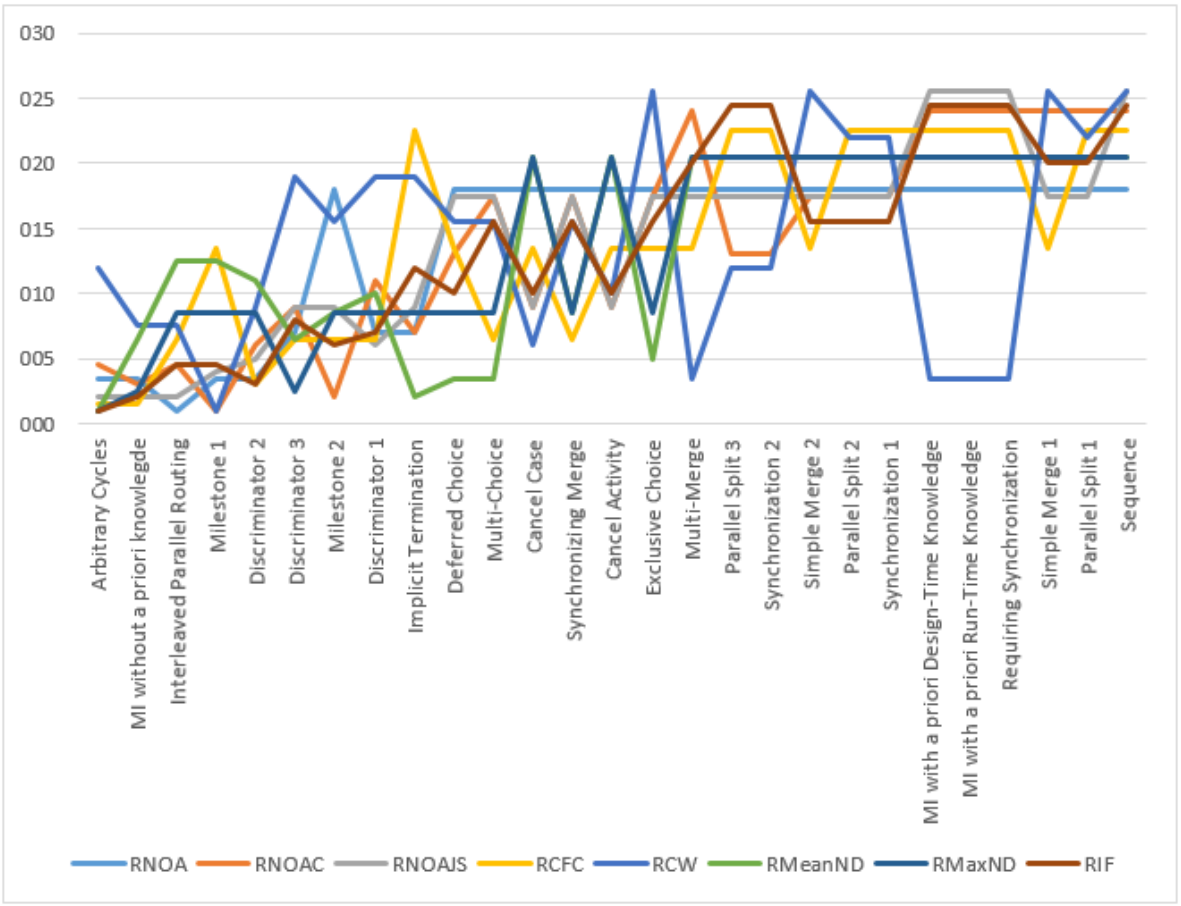


Fig. 39: Pattern comparison (ranks)

Another ground for comparison can be found by ranking the patterns. This is done by assigning mean weighted ranks to the different patterns according to their measured complexity. The results of this comparison can be found in Fig. 39. The patterns are ranked according to their average ranks for the various metrics. This means that the most complex pattern is ranked as 1. The ranks are weighted by their means. The different metrics follow a similar trend, which could be seen as reductions in complexity (from left to right). However some metrics seem to deviate from the trend. Especially the CW, CFC and MeanND rankings show obvious deviations from the general trend. This indicates that these metrics might measure different aspects of complexity in comparison with the other metrics.

The Arbitrary Cycles, MI without a priori Knowledge, Interleaved Parallel Routing, and Milestone patterns seem to be the most complex according to the metrics discussed. This was to be expected, because these patterns also required the most explanation in the previous section (4.2 Workflow Patterns). However, the CW metric deviates from this ascertainment. Instead, it seems to judge the MI patterns with a priori Knowledge and Requiring Synchronization as more complex, while the other metrics seem not to. The explanation for this can found in the weight that is assigned to multiple instance activity by the metric, while other metrics do not seem to address this complexity.

4.3.4 Conclusions

By applying factor analysis we identified that the metrics seem to measure two different aspects of complexity. These two aspects were identified as *structural complexity* and *functional complexity*. The structural complexity metrics identified are the NOAJS, CFC, MeanND, MaxND and IF. These metrics are mostly related to control flows and structure. The functional complexity metrics identified are NOA, NOAC and CW. These metrics are not as much linked to control flow, but rather to the nature of the elements within the BPM. This answers our research question with regards to whether the metrics measure the same thing.

By computing the Spearman Ranked Correlation Matrix we have identified that the different metrics seem to (overall) rank the patterns in the same way with regards to their complexity. There are however some interesting deviations: IF and CW do not seem to rank the patterns in the same way which could be grounds for further research.

There are problems with the interpretation of individual metrics. It is hard to assign a qualitative categorization of 'complexity' to specific values of the metrics. It seems intuitive though that a higher value indicates higher complexity (and thus lower understandability), however the question remains

what the difference is between slightly, moderately or heavily complex. This is however a dangerous assumption. Some representations in BPMN (for example Parallel Split 1 and 2) might appear confusing to the user, because even though there are fewer elements in the model, not all the information with regards to instances and synchronization are explicitly stated. To make things worse, the metrics do not seem to be equally sensitive to the complexity of the patterns analyzed. By using weighted values and mean ranked complexity it was found that the metrics do not express the same sensitivity. This was to be expected, because the metrics differ in their computational nature. For example the IF metric is highly sensitive due to its quadratic nature.

According to most metrics, the highly complex patterns seem to be the Arbitrary Cycles, MI without a priori Knowledge, Interleaved Parallel Routing and Milestone. This was in line with expectations, because these patterns' functionality also required the most explanation earlier this chapter. The ranking according to the CW metric however deviates from the general trend. This is explained by its penalizing nature for multiple instance activity.

By answering the several research questions proposed in the first chapter, an answer can be formulated to the general research question. Applying just a single metric from the metrics discussed does not seem to be the preferable way to measure a BPM's complexity and proxy for its understandability. The reason for this is because the metrics seem to measure different aspects of complexity. It could be that a BPM might be considered complex by the mean nesting depth metric, but considered the opposite by the Control Flow Complexity metric, as is demonstrated by the Implicit Termination pattern. A combination of metrics therefore seems to be the desirable path to follow. From the factor analysis it can be concluded that combining metrics that either measure *structural* or *functional complexity* seems to be the best way to go. Good metrics for structural complexity seem to be the NOAJS, CFC, MeanND, MaxND and IF metrics. For measuring functional complexity, NOA, NOAC and CW seem to be the best metrics.

Chapter 5: Conclusions

5.1 Concluding remarks

Keeping control over process complexity and thus understandability is important to keep the model understandable and in turn usable for users. For a business it is important that all the stakeholders can participate throughout all the phases of business process design, both of the present situation and for the future. In order to improve the communication of BPM, this study compares the combinations of different metrics to improve BPM understandability.

BPM understandability has been related in the literature to many theories derived from social sciences. Cognitive Load Theory, the SEQUAL model, Semiotic Theory, Resource Allocation Theory and Cognitive Theory of Multimedia Learning all provide interesting perspectives with regards to BPM understandability and their implications have been reviewed. Cognitive Load Theory implies that when more elements are added to the BPM, it should become more difficult to understand. Not only the number of elements impacts the understandability, but also the meaning that is given to the elements has its influence. Semiotic theory addresses this 'ladder' of understanding by decomposing understandability by syntax, semantics and pragmatics. The focus within this master's thesis is however on the syntactical level.

Different metrics for assessing the syntactical level of complexity of BPM have been discussed and their computational aspects have been explained. There are however other factors identified outside of those defined by the BPM alone. Resource Allocation Theory suggests that the user of the model should be able to understand a BPM better if he has more experience working with them. The literature therefore stresses the need for education in order to increase the understanding of more complex models. This is confirmed by the Cognitive Theory of Multimedia Learning, which suggests that not only the experience of working with BPM influences the understanding of the user, but also the way in which the BPM is presented. CTML can also be linked to semiotic theory: both define subsequent levels of understanding which can be matched together. Syntactical and semantical understanding can be linked to surface understanding. In order to acquire deep understanding the user needs to understand the BPM at the pragmatical level.

Due to the close relationship between complexity and understandability of BPM, some popular metrics for process complexity are discussed. These metrics form a close proxy for determining a BPM's level of understandability. The metrics do not seem to be very useful on an individual basis: they seem to

measure different aspects of a BPM's complexity. This assumption is tested by applying the various metrics to Van Der Aalst's (2003) Workflow Patterns (WFP). The data from these metrics is then analyzed with factor analysis and the different patterns are compared. By factor analysis, two dimensions of complexity were identified that were measured by the metrics. This seems to confirm our initial thought that the metrics measure different aspects of complexity. These two dimensions were identified as addressing *structural complexity* and *functional complexity*. Metrics that seem to target structural complexity are NOAJS, CFC, MeanND, MaxND and IF. NOA, NOAC and CW however seem to target the functional complexity of a BPM. The metrics do however seem to rank the patterns in the same way with regards to their complexity. The only exception is the CW metric, which could provide grounds for further research. In addition, the metrics do not seem to be equally sensitive, which was expected because the metrics differ in their computational nature. For example the IF metric is very sensitive due to its quadratic form. The patterns that were identified by the various metrics weighted ranking as being the most complex were also the patterns that needed the most explanation of their functionality. The CW metric did however assign different patterns higher complexity levels because of its penalizing nature for multiple instance activity.

All of this results in an answer to the general research question posed by this master's thesis: what are useful metrics or combinations of metrics for assessing business process model understandability? It can be concluded that a single metric from the ones discussed does not cover everything. The reason for this is because the metrics seem to measure different aspects of complexity. It is therefore preferred to use combinations of metrics that on the one side measure structural complexity and functional complexity on the other side.

5.2 Guidelines for future research

Most academic research with regards to understandability of BPM has been directed towards control flows and layout characteristics. However, not much research has been conducted (to my knowledge) about other aspects of a BPM that can influence understandability. BPM are characterized by their multi-dimensional nature. They not only give information about different activities, but also who is responsible or accountable for certain activities and what resources need to be addressed. So far, I have not come across much literature that addresses either the way responsibility/accountability or the allocation of resources is demonstrated in a BPM and how they affect the understandability of models.

With regards to the cognitive metrics on the syntax level, the knot count metric might be an interesting metric for further empirical research. The knot count could be an indication for the 'well-structuredness' and therefore the understandability of a BPM (W. M. Van der Aalst, 1998). Combining the Control-Flow Complexity metric and the Nesting Depth metrics could provide interesting results, as both cognitive weights with regards to structure and depth are measured. Maybe other perspectives can be offered by assigning scores to anti-patterns, or implementing detection mechanisms for this group of patterns in design tools for BPM. The Information Flow metric as introduced by Sun and Hou (2014) also offers interesting perspectives for assessing BPM complexity and understandability.

In this master's thesis Workflow Patterns are measured in terms of their understandability on the syntax level. However, on the semantics and pragmatics level not much research has been done to my knowledge. A few steps have been made in the semantics direction by Mendling (2010), but for now the amount of literature on these subjects seems to be limited. The reason for this could be that the preceding level, according to semiotic theory, syntax has not matured enough yet in BPM research.

References

Aalst, W. M. P. van der, Hofstede, A. H. M. ter, Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, *14*(1), 5–51. http://doi.org/10.1023/A:1022883727209

Abd Ghani, A. A., Koh, T. W., Muketha, G. M., & Wong, P. W. (2008). Complexity metrics for measuring the understandability and maintainability of Business Process Models using Goal-Question-Metric (GQM). *International Journal of Computer Science and Network Security*, *8*(5), 219–225.

Aguilar, E. R., García, F., Ruiz, F., & Piattini, M. (2007). An Exploratory Experiment to Validate Measures for Business Process Models. In *RCIS* (pp. 271–280).

Bandara, W., Gable, G. G., & Rosemann, M. (2007). Critical success factors of business process modeling. Geraadpleegd van http://eprints.qut.edu.au/8755/1/8755.pdf

Biggerstaff, T. J., Mitbander, B. G., & Webster, D. E. (1994). Program Understanding and the Concept Assignment Problem. *Commun. ACM*, *37*(5), 72–82. http://doi.org/10.1145/175290.175300

Burton-Jones, A., & Meso, P. (2008). The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. *Journal of the Association for Information Systems*, *9*(12), 1.

Burton-Jones, A., Wand, Y., & Weber, R. (2009). Guidelines for empirical evaluations of conceptual modeling grammars. *Journal of the Association for Information Systems*, *10*(6), 1.

Cardoso, J. (2005). Control-flow complexity measurement of processes and Weyuker's properties. In *6th International Enformatika Conference* (Vol. 8, pp. 213–218). Geraadpleegd van http://www.waset.org/publications/11011

Cardoso, J., Mendling, J., Neumann, G., & Reijers, H. A. (2006). A discourse on complexity of process models. In *Business process management workshops* (pp. 117–128). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/11837862_13

Carliss, Y., Baldwin, C. Y., & Clark, B. (1997). Managing in the age of modularity. *Harvard Business Review*, *75*(5), 84–93.

Cruz-Lemus, J. A., Maes, A., Genero, M., Poels, G., & Piattini, M. (2010). The impact of structural complexity on the understandability of UML statechart diagrams. *Information Sciences*, *180*(11), 2209–2220. http://doi.org/10.1016/j.ins.2010.01.026

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, *58*(3), 358–380.

Eriksson, H.-E., & Penker, M. (1998). *Business Modeling With UML: Business Patterns at Work* (1st ed.). New York, NY, USA: John Wiley & Sons, Inc.

Figl, K., & Laue, R. (2011). Cognitive Complexity in Business Process Modeling. In H. Mouratidis & C. Rolland (Red.), *Advanced Information Systems Engineering* (pp. 452–466). Springer Berlin Heidelberg.

Figl, K., Mendling, J., & Strembeck, M. (2013). The influence of notational deficiencies on process model comprehension. *Journal of the Association for Information Systems*, *14*(6), 312–338.

Figl, K., & Strembeck, M. (2014). On the Importance of Flow Direction in Business Process Models. Geraadpleegd van http://wwwi.wu-wien.ac.at/home/mark/publications/icsoft-ea14.pdf

Genon, N., Heymans, P., & Amyot, D. (2011). Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In *Proceedings of the Third International Conference on Software Language Engineering* (pp. 377–396). Berlin, Heidelberg: Springer-Verlag. Geraadpleegd van http://dl.acm.org/citation.cfm?id=1964571.1964605

Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: a "cognitive dimensions" framework. *Journal of Visual Languages & Computing*, *7*(2), 131–174.

Gruhn, V., & Laue, R. (2006a). Adopting the cognitive complexity measure for business process models. In *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on* (Vol. 1, pp. 236–241). IEEE. Geraadpleegd van http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4216417

Gruhn, V., & Laue, R. (2006b). Complexity metrics for business process models. In *9th international conference on business information systems (BIS 2006)* (Vol. 85, pp. 1–12). Citeseer. Geraadpleegd van http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.2111&rep=rep1&type=pdf

Gruhn, V., & Laue, R. (2009). Reducing the cognitive complexity of business process models (pp. 339–345). IEEE. http://doi.org/10.1109/COGINF.2009.5250717

Gustafsson, J. (2000). Metrics calculation in MAISA. *University of Helsinki, Department of Computer Science*. Geraadpleegd van http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.8954&rep=rep1&type=pdf

Hahn, J., & Kim, J. (1999). Why are some diagrams easier to work with? Effects of diagrammatic representation on the cognitive intergration process of systems analysis and design. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *6*(3), 181–213.

Henry, S., & Kafura, D. (1981). Software structure metrics based on information flow. *Software Engineering, IEEE Transactions on*, (5), 510–518.

Holl, A., & Valentin, G. (2004). Structured business process modeling (SBPM). *Information systems research in Scandinavia (IRIS 27)(CD-ROM)*. Geraadpleegd van http://www.informatik.fh-nuernberg.de/professors/Holl/Personal/sbpm_IRIS_2004.pdf

Ince, D. C., & Shepperd, M. J. (1989). An empirical and theoretical analysis of an information flow-based system design metric. In *ESEC'89* (pp. 86–99). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/3-540-51635-2_34

Kanfer, R., Ackerman, P. L., Murtha, T. C., Dugdale, B., & Nelson, L. (1994). Goal setting, conditions of practice, and task performance: A resource allocation perspective. *Journal of Applied Psychology*, *79*(6), 826.

Kiepuszewski, B., ter Hofstede, A. H. M., & Bussler, C. J. (2000). On structured workflow modelling. In *Advanced Information Systems Engineering* (pp. 431–445). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/3-540-45140-4_29
Koschmider, A., & Reijers, H. A. (2015). Improving the process of process modelling by the use of domain process patterns. *Enterprise Information Systems*, *9*(1), 29–57.
Krogstie, J., Sindre, G., & Jørgensen, H. avard. (2006). Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, *15*(1), 91–102.
Laue, R., & Awad, A. (2010). Visualization of business process modeling anti patterns. *Electronic Communications of the EASST*, *25*. Geraadpleegd van http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/344
Lindland, O. I., Sindre, G., & Solvberg, A. (1994). Understanding quality in conceptual modeling. *Software, IEEE*, *11*(2), 42–49.
List, B., & Korherr, B. (2006). An Evaluation of Conceptual Business Process Modelling Languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing* (pp. 1532–1539). New York, NY, USA: ACM. http://doi.org/10.1145/1141277.1141633
Liu, R., & Kumar, A. (2005). An analysis and taxonomy of unstructured workflows. In *Business Process Management* (pp. 268–284). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/11538394_18
Mayer, R. E. (2002). Multimedia learning. *Psychology of Learning and Motivation*, *41*, 85–139.
McCabe, T., J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, *2*(4). http://doi.org/10.1109/TSE.1976.233837
Melcher, J., Mendling, J., Reijers, H. A., & Seese, D. (2010). On Measuring the Understandability of Process Models. In S. Rinderle-Ma, S. Sadiq, & F. Leymann (Red.), *Business Process Management Workshops* (pp. 465–476). Springer Berlin Heidelberg. Geraadpleegd van http://link.springer.com/chapter/10.1007/978-3-642-12186-9_44
Mendling, J., Reijers, H. A., & Recker, J. (2010). Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, *35*(4), 467–482.
Mendling, J., & Strembeck, M. (2008). Influence factors of understanding business process models. In *Business information systems* (pp. 142–153). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/978-3-540-79396-0_13
Mendling, J., Strembeck, M., & Recker, J. (2012). Factors of process model comprehension—findings from a series of experiments. *Decision Support Systems*, *53*(1), 195–206.
Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580. http://doi.org/10.1109/5.24143
Nordbotten, J. C., & Crosby, M. E. (1999). The effect of graphic style on data model interpretation. *Information Systems Journal*, *9*(2), 139–155. http://doi.org/10.1046/j.1365-2575.1999.00052.x
N. Russell, A. H. M. ter H. (2006). Workflow Control-Flow Patterns: A Revised View.
Paas, F., & Ayres, P. (2014). Cognitive Load Theory: A Broader View on the Role of Memory in Learning and Education. *Educational Psychology Review*, (2), 191.
Petre, M. (1995). Why looking isn't always seeing: readership skills and graphical programming. *Communications of the ACM*, *38*(6), 33–44.
Recker, J., Rosemann, M., Indulska, M., & Green, P. (2009). Business process modeling-a comparative analysis. *Journal of the Association for Information Systems*, *10*(4), 1.
Reijers, H. A., & Mendling, J. (2011). A Study Into the Factors That Influence the Understandability of Business Process Models. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *41*(3), 449–462. http://doi.org/10.1109/TSMCA.2010.2087017
Reijers, H. A., Mendling, J., & Dijkman, R. M. (2011). Human and automatic modularizations of process models to enhance their comprehension. *Information Systems*, *36*(5), 881–897. http://doi.org/10.1016/j.is.2011.03.003
Reijers, H., & Mendling, J. (2008). Modularity in process models: Review and effects. In *Business Process Management* (pp. 20–35). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/978-3-540-85758-7_5
Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, *33*(1), 64–95. http://doi.org/10.1016/j.is.2007.07.001
Scheer, A.-W. (2000). *ARIS — Business Process Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg. Geraadpleegd van http://link.springer.com/10.1007/978-3-642-57108-4
Schrepfer, M., Wolf, J., Mendling, J., & Reijers, H. A. (2009). The impact of secondary notation on process model understanding. In *The Practice of Enterprise Modeling* (pp. 161–175). Springer. Geraadpleegd van http://link.springer.com/chapter/10.1007/978-3-642-05352-8_13
Schroeder, A. (1984). Integrated program measurement and documentation tools. In *Proceedings of the 7th international conference on Software engineering* (pp. 304–313). IEEE Press. Geraadpleegd van http://dl.acm.org/citation.cfm?id=801985
Shao, J., & Wang, Y. (2003). A new measure of software complexity based on cognitive weights. *Electrical and Computer Engineering, Canadian Journal of*, *28*(2), 69–74.
Sun, H., & Hou, H. (2014). Study on Complexity Metrics of Business Process. Geraadpleegd van http://www.atlantis-press.com/php/download_paper.php?id=12706
Sweller, J., & Chandler, P. (1994). Why some material is difficult to learn. *Cognition and instruction*, *12*(3), 185–233.
Thom, L. H., Reichert, M., & Iochpe, C. (2009). Activity patterns in process-aware information systems: Basic concepts and empirical evidence. *International Journal of Business Process Integration and Management*, *4*(2), 93–110.

Van der Aalst, W. M. (1998). The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, *8*(01), 21–66.
Van der Aalst, W. M. P. (2011). *Process mining discovery, conformance and enhancement of business processes*. Berlin Springer.
Ware, C., Purchase, H., Colpoys, L., & McGill, M. (2002). Cognitive measurements of graph aesthetics. *Information Visualization*, *1*(2), 103–110.
Weyuker, E. J. (1988). Evaluating software complexity measures. *Software Engineering, IEEE Transactions on*, *14*(9), 1357–1365.
White, S. A. (2004). Process modeling notations and workflow patterns. *Workflow handbook*, *2004*, 265–294.
White, S. A. (2008). *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc.
Wohed, P., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H., & Russell, N. (2005). Pattern-based Analysis of BPMN. Geraadpleegd van http://eprints.qut.edu.au/archive/00002977

Appendix

Introduction to BPMN

BPMN was originally developed by the Business Process Management Initiative (BPMI) to support the graphical representation of business processes. The objective of the standard is to improve the communication of business processes between all the stakeholders. It should be a communicational tool that on the one side can be easily understood, but on the other should offer the technical functionality to express very complex business processes. BPMN is only meant to model actual processes and it cannot be used to make organizational diagrams or data models. The graphical notation shows a lot of similarities with the Activity Diagrams from Unified Modeling Language (UML AD), however the target audience of both languages is different. While UML AD is mainly aimed at software engineers, BPMN has been created for business users.

A brief overview of the elements of BPMN can be found in Fig. 40. These elements are briefly discussed at the individual level. For their exact and broader specifications the reader is referred to the website of the Object Management Group[4].

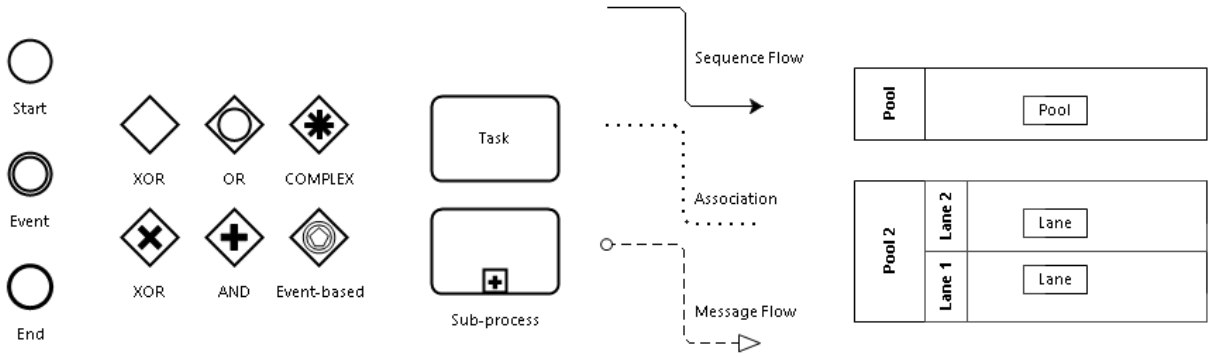


Fig. 40: Elements of BPMN 2.0

Events

Start events

The none-start event symbolizes the main start of the business process. This is where the flow starts: a token is generated that will travel through the rest of the BPM. A token is a theoretical concept and can be seen as an instance of the process. This token then travels through the BPM following the sequence flows, demonstrating the flow of the process in reality. Multiple tokens or instances can be generated by Start events or other elements of the BPM. Start events are illustrated as circles with

[4] The specifications for the BPMN 2.0 standard can be found at: http://www.omg.org/spec/BPMN/index.htm

optionally a symbol inside. Other variations of the start event, as can be seen in Fig. 41, are triggered when a message is received or a condition is satisfied. The graphical representations are fairly straightforward and can be found in Fig. 41. There are other kinds of start events possible within BPMN 2.0, however they are not relevant for understanding the examples given in this thesis.

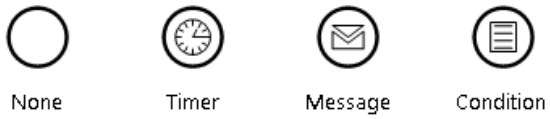


Fig. 41: Start events

End events

The process ends when all tokens in the BPM have reached an end-event and are thus destroyed. All the possible variations of end-events are listed in Fig. 42. End events are illustrated by a thicker circle than start events. Other variations include end-events that trigger activities such as sending a message or signaling other parts of the process. The Terminate-event kills all remaining tokens in the BPM and ends the process.

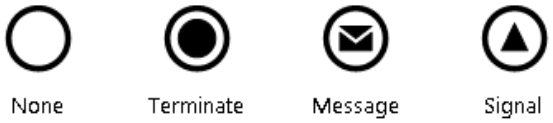


Fig. 42: End events

Intermediate events

Intermediate events are neither start- nor end-events. They trigger activities, usually during or in between task execution. Intermediate events are symbolized by a double circle with possible addition of a symbol inside. These events show similarities with the corresponding symbols of start- and end-events. The different variations are listed in Fig. 43.

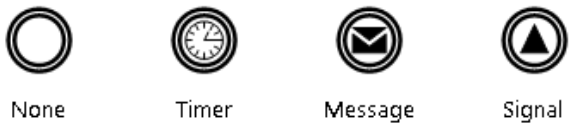


Fig. 43: Intermediate events

Tasks

Tasks are the main activities in the BPM. They either describe tasks executed by a user, a service or scripts. A user task is a task that has to be done by a person. A service task is a task that is executed by an application or web service, for example a server query. A script task is a task that is executed by a script in the business process engine. Usually these scripts are written in a programming language like JavaScript. The visual representation of these tasks is given by a rounded rectangle

with a short textual description of the activity inside. Small icons in the upper left side of the rectangle show whether the task is done by a user, a service or a script, or that they include sending or receiving messages. Within the examples given in this master's thesis we assume that all tasks are none-tasks, which do not have an icon. These activities can be looped with added symbols at the bottom side of the rounded rectangle. Examples of the graphical representations can be found in Fig. 44 and Fig. 45.

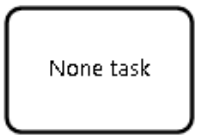


Fig. 44: Tasks

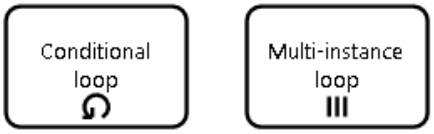


Fig. 45: Looping tasks

Sub-processes

Sub-processes are parts of the BPM that describe a 'chunk' of the BPM that can be 'grouped' as a single task. This grouping is done to lower the granularity of the BPM and increase the overall understandability of the model, or to allow for specific functionality that would not be possible within the normal control flow. Some examples of these exceptional situations are shown in the next chapter. The graphical representation of such a sub-process or sub-task can be given in either its collapsed or expanded form. When the sub-process is collapsed, it looks very similar to a normal task however a plus-sign is added at the bottom edge of the rounded rectangle and the label is written outside of the element. When the sub-process is expanded, the underlying sub-process is visible to the reader. Examples of the collapsed and expanded views of sub-processes are given in Fig. 46.

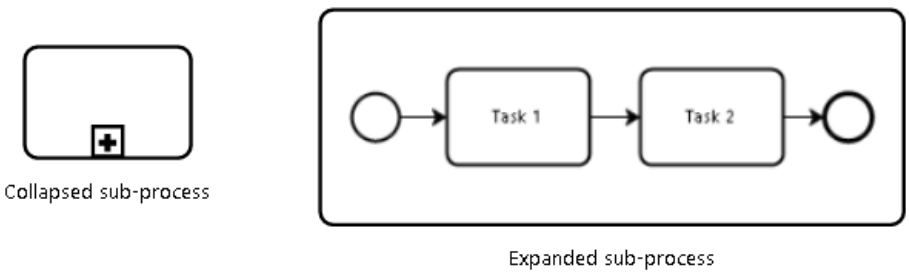


Fig. 46: Sub-processes

Gateways

BPMN uses gateways to split or merge control flows. Gateways are used to express the sequence flow of the process and this way they define the behavior of the process. Gateways in BPMN are visualized as a slightly rotated (45 degrees) square. The symbol (or absence thereof) within the square gives the logical implications of the construct. The different types of gateways (AND, XOR, OR, event-based and complex) are described and illustrated below.

AND-gateway

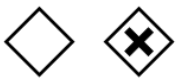
The split of the AND-gateway instructs the flow of the process to continue in all outgoing branches of the gateway. This means that multiple tokens are created in the remainder of the BPM that might eventually be synchronized by a merging AND-gateway. The AND-gateway is used when multiple tasks need to be done in parallel, for example by different actors.

AND

Fig. 47: AND-gateway

XOR-gateway

The XOR-split-gateway implies an exclusive choice for the BPM. Only *one* of many possible flows will be chosen based on conditions specified either inside the gateway or on the branches that follow from it. This means that the different conditions that are connected to the gateway need to be mutually exclusive, only one single condition should be met and there should be alternatives. The matching XOR-merge-gateway however allows all incoming branches to continue in the flow following the gateway. This would mean if the AND-split-gateway would be followed by a XOR-join-gateway, multiple tokens will remain in the process and all following activities might be executed multiple times. The XOR-gateway can be represented in two ways in BPMN. It can either have an X inside, or it can be left blank. Both representations have the same meaning[5]. Examples are given in Fig. 48.



XOR XOR

Fig. 48: XOR-gateway

[5] To avoid confusion within this document only the variant *with* the X-symbol is used.

OR-gateway

The OR-gateway differs from the XOR-gateway in such a way that multiple branches can be followed by the flow of the BPM, as long as the conditions given are met. This makes the interpretation of the OR-gateway perhaps a bit more complex than the XOR-gateway, especially since the merging happens differently. The OR-gateway synchronizes all the incoming sequence flows and is able to identify which paths were activated by its splitting counterpart. An example of such a gateway is given in Fig. 49.

Fig. 49: OR-gateway

Event-based gateways

Event-based gateways have flow-handling capacity that differs greatly from the other gateways that have been discussed so far. Event-based gateways use a 'pull' mechanism to decide on what control flow path should be followed next. Following on the gateways are different kinds of events: waiting for a message, waiting for a timer to expire, … Whichever event is triggered first, that path is followed by the control flow. The symbols for event-based gateways are demonstrated in Fig. 50.

Fig. 50: Event-based gateways

Because this behavior might be confusing at first, an example of an event-based gateway can be found in Fig. 51. In this example, after task A is completed the control flow continues to the Event-based gateway. The token then waits for either a message received or a timer that expires. If the timer expires before a message is received, task C is started. If the message arrives after the timer has expired (et vice versa) nothing happens, the flow has already continued.

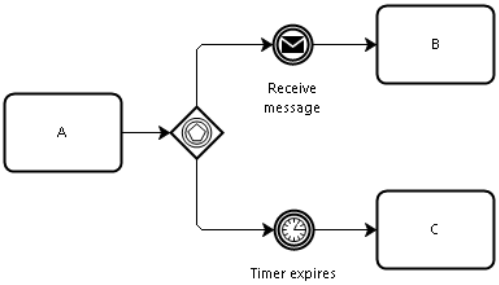


Fig. 51: Example Event-based gateway

Complex gateway

The Complex gateway allows for the specification of more complex logic than is allowed by the other gateways discussed above. For example, a merging Complex-gateway allows for a logic that N out of M control flow paths should arrive at the gateway before the continuation of the control flow is allowed. The graphical representation of such a gateway is given in Fig. 52.

Fig. 52: Complex gateways

Flows and associations

The *control flow* of the process in BPMN is illustrated by a solid arrow, with the arrow pointing in the direction of the flow. Associations are illustrated with a dotted line without arrowheads. Associations are made for example between extra explanatory labels that give deeper meaning to specific elements of the BPM. Message flows, for example between departments or parties, are indicated by a dashed line with arrows pointing in the direction of the flow. They do not symbolize process control flow, but indicate communicational flows and symbolize extra information for the reader of the BPM.

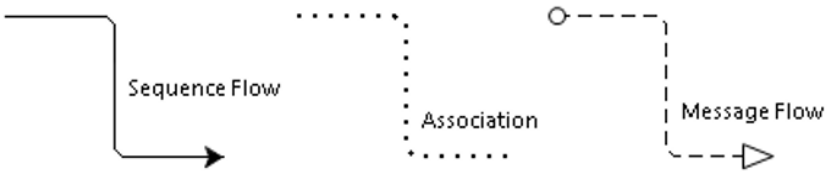


Fig. 53: Flows and association

Pools and lanes

Within BPMN, pools and lanes are used to demonstrate the differences between actors or parties in the BPM. Different lanes are different actors within the process and give some indication of responsibilities of specific tasks. Their positioning inside a lane indicates that the actor that is associated with that lane is responsible for the execution of that activity. Different pools indicate external parties, for example suppliers or customers. They are outside of the process and the process flow can interact with these parties through message flows. The graphical representation of a pool and its lanes is given in Fig. 54.

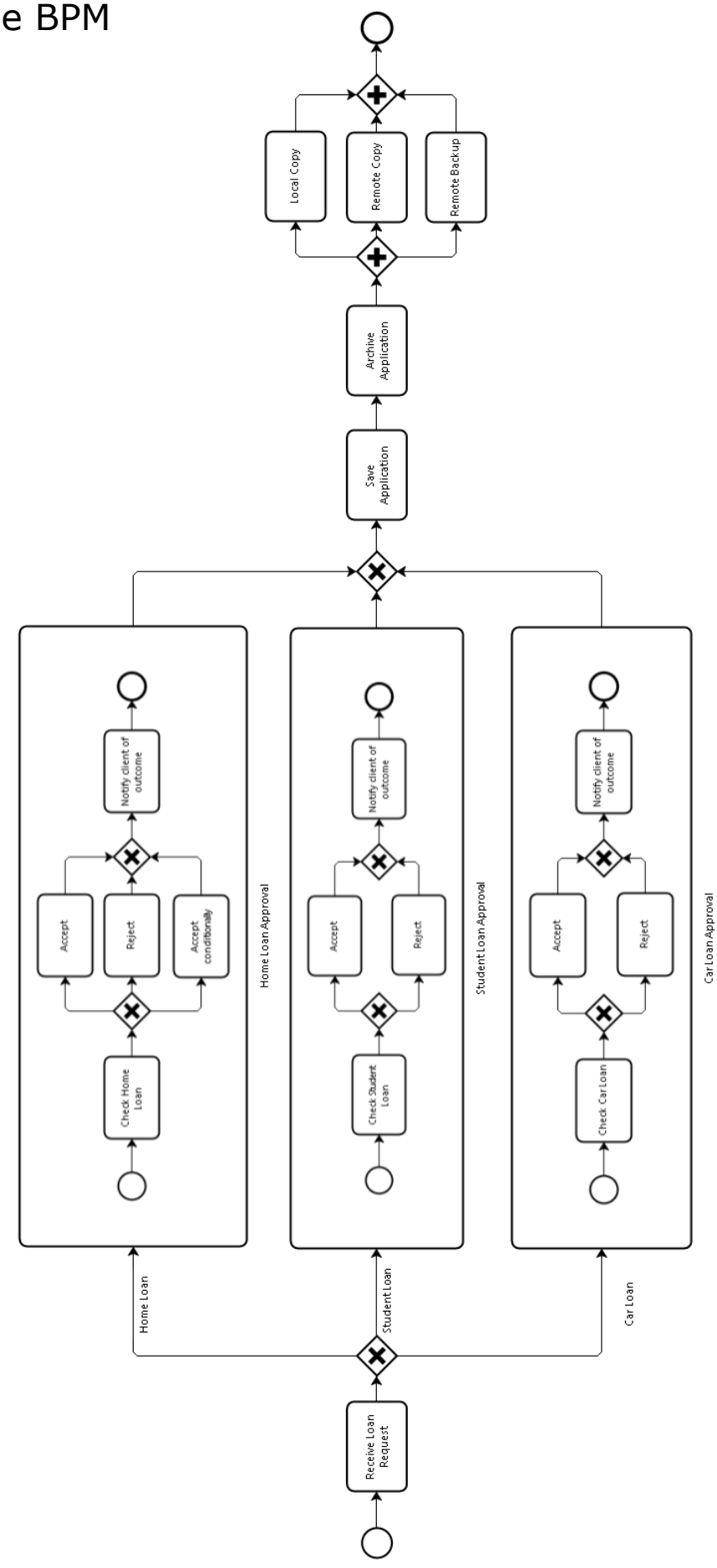
Fig. 54: Pools and lanes

Concluding remarks

This short summarization is meant as a short introduction to BPMN. It does however contain far from all of the specifications that compose the BPMN standard. The information as given in this preliminary section should be sufficient for comprehending the examples given in this document. For the exact and complete specifications the reader is referred to the official specifications[6] on the website of the Object Management Group.

[6] The full specifications for the BPMN 2.0 standard can be found at: http://www.omg.org/spec/BPMN/index.htm

Example BPM


Receive Loan Request
Home Loan
Student Loan
Car Loan
Check Home Loan
Accept
Reject
Accept conditionally
Notify client of outcome
Home Loan Approval
Check Student Loan
Accept
Reject
Notify client of outcome
Student Loan Approval
Check Car Loan
Accept
Reject
Notify client of outcome
Car Loan Approval
Save Application
Archive Application
Local Copy
Remote Copy
Remote Backup

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Comprehension of Business Process Models: An evaluation of metrics for understandability

Richting: **Master of Management-Management Information Systems**
Jaar: **2015**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Van Eijk, Koen

Datum: **31/05/2015**