# Business Process Modeling

as a means to bridge

# The Business-IT Divide

Martin Devillers

Master's Thesis

Information Sciences


Radboud University Nijmegen

*Supervisors*
Dr. Stijn Hoppenbrouwers
Hidde Andriessen
Gert Jan Timmerman

Thesis Number 156IK
Martin Devillers
Version 1.3
Aug 2011

“All models are wrong, some are useful”

George E.P. Box

## Preface

This Master's thesis is the result of six months of research performed by me, Martin Devillers, as part of my internship at Info Support and my thesis for the Information Sciences Master of the Radboud University Nijmegen.

I would like to extend a word of thanks to the various organizations and people who in some manner have made a contribution to my research. I would like to thank the following organizations:

- *Radboud University Nijmegen*, for providing me with an adequate education, thus granting me with a vast amount of theoretical knowledge, which formed the bedrock of my research.
- *Info Support*, for accepting me as an intern and providing the appropriate resources required by me to perform my research.

I would like to thank three people for their involvement in my endeavors:

- *Stijn Hoppenbrouwers*, my supervisor at the Radboud University Nijmegen, for his guidance during the research project. His insight into conceptual modeling and group model building helped me considerably in my research.
- *Hidde Andriessen*, my principal at Info Support, for providing the original assignment and the case which instigated my research. Contrary to what I expected, Hidde was more interested in the theoretical subjects underlying his case, than an actual solution. This motivated me to take a highly theoretical and explorative approach towards my research.
- *Gert Jan Timmerman*, my technical supervisor at Info Support, for his excellent feedback and the numerous hours we spent on discussions. His perspectives on language and knowledge management helped me considerably in my research. But perhaps more importantly, meeting Gert Jan has taught me that commercial organizations can too be a place for scientists.

Lastly, I would like to reflect on the research process itself. Clearly, the focus of my attention with respect to the research topic shifted considerably over the six months. It would appear that as my knowledge of business process modeling and related topics expanded, the things I would write one day would seem wrong or irrelevant the next. Due to my perfectionist nature, I was not content to write anything I did not fully stand by nor did not entirely understand. As a result, I spent a significant time reading papers and writing text only to end up discarding my results. While at the time I found this lack of progress to be rather dissatisfying, I now realize that it was necessary and in all likelihood unavoidable. Although I do not intend to join the research community after I have finished my masters, I have gained an appreciation for the struggle scientists undergo in a world where there is vastly more knowledge available than any man could ever comprehend.

**Abstract**

A great number of business process modeling endeavors aim to create representations of business processes which can be translated to computer software by IT stakeholders while remaining understandable to business stakeholders. However in practice, business process models rarely meet these high demands, leading either to models which are too informal to be interpreted by a computer or to models which are incomprehensible to business stakeholders. Our research attempts to assess why these problems occur and whether it is fundamentally feasible to create models that meet these demands.

Although the topic of business process modeling has received great attention from the research community, most of this attention is focused on the formal aspects of these models, such as computer simulation or mathematical properties. In contrast, little research addresses the informal aspects of these models, such as their accuracy in representing the real world business processes they are based on. We argue that these informal aspects play a pivotal role in creating a shared understanding of business processes among business and IT stakeholders.

We performed a two phase research in order to answer the question whether business process modeling can serve as a common language between business and IT stakeholders. The first phase is of a philosophical nature as it addresses a wide range of theories. These include the historic business-IT alignment problem, theories on language, conceptual modeling and generic system design principles. The second phase is of an empirical nature as it focuses on a case study of a real world BPM project. The study reveals a series of problems which we subsequently generalized using our theories gathered during the first phase. This allows us to not only produce a list of generic issues currently affecting BPM projects, but also to explain why these issues occur and how they can be avoided.

Most notably, our research reveals how the different manner in which business and IT stakeholders create and interpret business process models can lead to misunderstandings. Additionally, our results show that the influence of the IT discipline on BPM actually inhibits the development of business friendly modeling languages. We argue that if BPM is to truly become a management discipline, the focus should be placed on the usability of process modeling for the business, rather than IT.

**Table of Contents**

**Table of Figures**

# 1 Introduction

Business process management (BPM) is an increasingly important topic for large organizations. Business process-awareness is at an all-time high and contemporary organizations invest considerable effort in BPM projects in order to attain enterprise operational efficiency [1]. Despite the great attention BPM has received both from the business world and the research community, organizations are still unable to fulfill their demand for expert knowledge in BPM projects. Both BPM projects and BPM as a research topic are associated with a degree of uncertainty and there exists much quarrel in the BPM community. To illustrate, at the time of writing there is still no generally accepted consensus on what BPM actually stands for or what it encompasses [2]. The lack of a solid theoretical foundation for BPM is one of the primary motivators for our research [3].

A core element of BPM is the so-called *business process model*[1], which is a systematic description of a business process generally in the form of a series of activities. According to Ould [4], there are three goals one tries to achieve with a business process model. To each goal, we assign a certain level of modeling effort, which we define as the time, methodological knowledge, technical support and stakeholder commitment required to achieve the desired goal. The three goals, in order from minimal modeling effort to maximal modeling effort are:

- *Description*, with the purpose of facilitating shared understanding amongst business stakeholders and to achieve consensus on the manner of which a business accomplishes its goals.
- *Analysis*, with the purpose of improving a business process in order to increase efficiency of the overall organization. Process models created for this purpose are generally measurable in some way (e.g. measuring the throughput of a production process or the average time required to complete an ordering process).
- *Enactment*, with the purpose of providing direct IT support to a business process. Process models that have been created for this purpose can either be directly executed by a process engine, translated into a machine executable model or used as a basis for software development [5].

While the first and second goal can certainly lead to organizational benefits, most BPM projects aim to achieve the third goal. In fact, the BPM approach originally focused on the automation of business processes. In this context, the inception of business process modeling can be seen as an attempt to create a common language between business and IT stakeholders. Our research focuses on whether this goal has been achieved, which also constitutes our main research question:

> *Can business process modeling serve as a common language between business and IT stakeholders?*

In order to answer this question, we conducted a two phase research. In the first phase, we examine the various theoretical principles underlying our main research question, while in the second phase we examine the problems currently present in BPM projects. The first phase of our research opens with an exploration of existing business process modeling languages:

> *What business process modeling languages currently exist, what are their origins and in what manner do they represent business processes?*

Answers to this question helps to shape the context of the problem we are attempting to solve by examining the current state of affairs.

[1] Be aware that the acronym BPM refers to business process *management* and not business process *modeling*.

Next, we give a short summary of the business IT alignment problem, although we assume that the reader already possess a basic understanding of this dilemma. We will make the assumption that the business IT alignment problem is essentially a language problem: Business stakeholders emphasize natural language, while IT stakeholders emphasize artificial language, thus hindering successful communication. With this assumption in mind, we attempt to answer the following question:

*How do natural language and artificial languages assist in the understanding and representation of business processes?*

Answers to this question aids in building an understanding of the language problem and assessing whether it is feasible to create a common language for business and IT stakeholders. The theory of languages is followed by the theory of conceptual modeling. These help to understand how the modeling activity shapes the business process:

*How does conceptual modeling assist in the design of business processes?*

The first phase of our research is concluded by a listing of system design principles and the application of these principles to the design of business processes. Moreover, it is shown that business process modeling can be used as a means to measure the successful application of these principles.

*What system design principles exist and can these be applied to the design of business processes?*

In the second phase of our research, we performed a case study in order to assess the problems in real world BPM projects. The empirical findings of the case study are combined with the theoretical findings of the first phase to produce a listing of generic issues that hinder business process modeling. Based on these issues, we draw our final conclusion on whether business process modeling has been successful in bridging the business-IT divide.

Answers to these questions will form the basis for a set of guidelines that business users can follow to create higher quality business process models. As these models are at the heart of most BPM projects, modeling improvements will consecutively benefit all stakeholders involved with the BPM project. Moreover, we hope to spread awareness of the fundamental issues associated with business process modeling in order to prevent future BPM projects from falling into common business process modeling pitfalls.

The remainder of this thesis is structured as follows: Chapter 2 provides a series of theories, which are subsequently applied to business process modeling in chapter 3. Chapter 4 provides a description and analysis of our case study, which lead to the listing of generic issues in chapter 5. To reiterate, chapter 2 and 3 represent the first phase of our research, while chapter 4 and 5 represent the second phase of our research. Chapter 6 provides the discussion of our research and chapter 7 provides the most prominent conclusions of our research.

# 2 Theoretical background

An elaboration of the various theoretical topics involved in our research will be provided in this chapter. The topics we will discuss are business process management, business process modeling, business-IT alignment, language theory, conceptual modeling, abstraction and system design principles.

## 2.1 *Business process management*

Business process management (BPM) is a systematic approach towards the definition, execution, management and refinement of business processes. A business process is a collection of activities or tasks that produce a specific service or product, generally involving both human interaction and computer applications.

All BPM activities can be attributed to one of the five phases of the BPM lifecycle:

- *Design phase*, during which existing business processes are identified and future business processes are designed. It is not uncommon for many stakeholders to be involved in this phase as business processes can be interdepartmental or even inter-organizational. Typical information required for the identification of business processes are tasks, quantifiable deliverables (e.g. documents), responsibilities, computer systems and required resources.
- *Modeling phase*, during which the information gathered during the design phase is made explicit in a business process model. These models are usually created with elaborate modeling tools, using a standard for business process modeling, such as the *Business Process Modeling Notation* (BPMN) and subsequently stored in a so-called modeling repository.
- *Execution phase*, during which computer applications are deployed to support the automation of the business process. In traditional organizations, there are many computer applications that each perform a specific task. These applications are not unaware of each other's existence nor are they aware of the context in which they are used. *Business process orchestration* attempts to connect these applications by a main application, which is aware of the business process [6]. In addition, more recent developments of BPM technology aim to use the business process model itself as a basis for automation. By formalizing the business process model, it can be interpreted and subsequently executed by a so-called *business process engine*. This approach allows computers to be aware of the actual business process as it is executed, which in turn enables a variety of advantages to the business.
- *Monitoring phase*, during which the performance of the implemented business processes is measured. The depth of the analysis depends on what BPM technologies were implemented during the execution phase. When a feature rich BPMS is used, one can measure process performance both at the global level and at the instance level. This information can subsequently be aggregated and displayed in comprehensive monitoring dashboards, which give managers quantifiable real-life data of the performance of their business. When this degree of rigor is employed during the monitoring phase, the act of monitoring becomes a goal by itself, which is also known as *business activity monitoring*.
- *Optimization phase*, during which business processes are optimized based on the findings of the monitoring phase. These optimizations may lead to the redesign of existing business processes or the design of additional business processes. As a result, the optimization phase can be the initiator of a new design phase, thus completing the BPM life-cycle.

The five phases of the BPM life-cycle shown in figure 1 form a never-ending loop, known as the *continuous process improvement* cycle. The modeling phase is, for obvious reasons, of great interest to our research, but another important phase is the design phase. As we will show, modeling can be a design activity by itself and designs tend to change when the modeling activity becomes involved.

In other words, design influence model and model influence design. Because of this, the distinction between what activities relate to the design of the business process and what relate to the modeling of the business process is not always clear. Indeed, alternative versions of the BPM life-cycle exist, which describe the design and modeling phase as one.

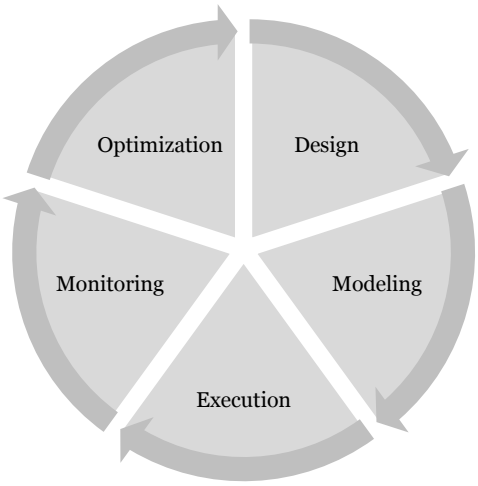


Figure 1 - BPM life-cycle

### 2.2 *Business process modeling*

Business process models play a pivotal role in the business process management discipline. There are many modeling methodologies available which can be used to model business processes. Some of these methodologies have been specifically designed for this purpose, while others predate the BPM discipline and have been adapted for this purpose. Regarding the three goals of process modeling (i.e. description, analysis and enactment), most methodologies tend to be suitable for only one of the three goals, but not all three.

The great diversity of business process modeling languages has already leaded to researchers attempting to chart all these languages [7]. Particularly, a research done by Hafedh Mili et al. gives an excellent overview of these languages [8]. We will provide a less elaborate overview and only describe those languages that have played an important role in business process modeling. The following languages will be covered: Flowchart, Petri net, Unified Modeling Language, Role Activity Diagram, Event-driven Process Chain, Integrated Definition for Function Modeling and Business Process Management Notation.

#### 2.2.1 Flowchart

A flowchart is a diagram that represents a process as a sequence of activities and decisions. Flowcharts are the oldest and most basic process related modeling methodology known, with their first reported occurrence dating back to the early twenties, where they were used by mechanical engineers to describe machine behavior.

Basic flowchart constructs are activities, decisions, start points and end points. These are the basic building blocks typically used to represent processes. More advanced flowcharts use data-flow constructs (e.g. documents or machine input/output) to denote what information flows throughout the process. Relationships in a flowchart are denoted by arrows which indicate a flow of control from one element to another. All elements in a flowchart are either directly or indirectly connected to one and another.

Figure 2 shows an example of a flowchart of a person watching TV. The process starts with the person turning on the TV, after which he observes the channel the TV is on. As long as the channel remains interesting, the person will continue to view that channel. When he is no longer interested in the channel, he can choose to switch the channel, granted there are any channels left which he has not seen. When all channels have been checked, the TV is turned off and the process ends.

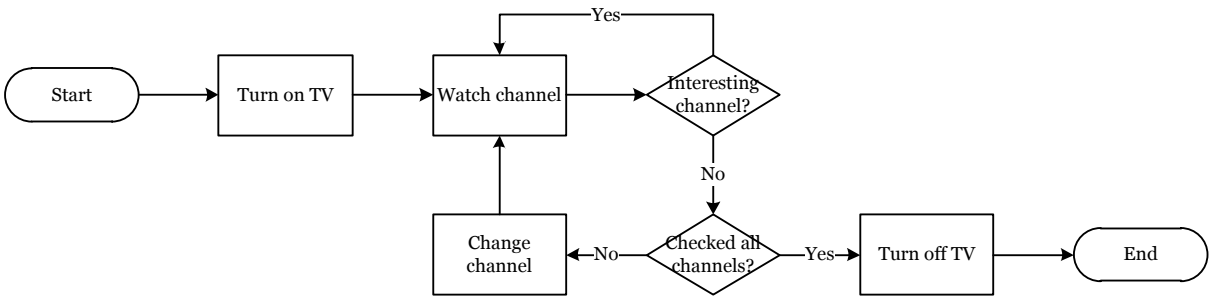



Figure 2 - Flowchart of a person watching TV

Besides the simple set of constructs, flowcharts offer little to none methodological support. There are some commonly agreed on best-practices for designing flowcharts (e.g. limit crossing arrows), but guidelines for the application of flowcharts differ greatly.

With respect to business process modeling, flowcharts are often used as a sketching tool during the earlier phases of the model development. The intuitiveness of flowcharts makes it an ideal tool for quickly charting processes in meetings or workshops.

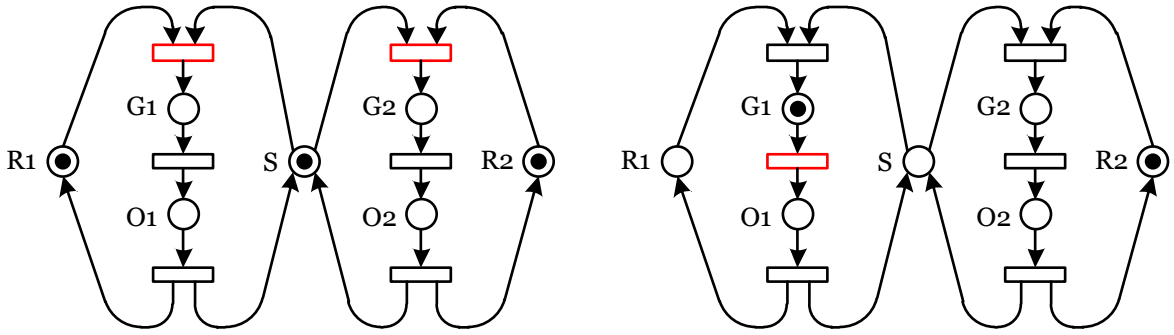
### 2.2.2 Petri net

A Petri net is a formal modeling language for the description of concurrent processes. Carl Adam Petri invented the graphical notation of Petri nets in 1939 for the purpose of describing chemical processes, while the concept of Petri nets as we know them today would not be introduced by him until the early sixties. Unlike other methodologies, Petri nets have a strict mathematical definition of their execution semantics, which means that all well-formed Petri nets can be interpreted and executed by a machine. Whereas most other methodologies focus on representing the structure of a process, Petri nets focuses on the actual behavior of a process. Moreover, the mathematical basis of Petri nets makes them suitable for various kinds of automated analysis.

A Petri net consists of places and transitions, which are connected by arrows, otherwise known as directed arcs. One place may have multiple arcs running to distinct transitions and one transition may have multiple arcs running to distinct places. However, no arcs may ever run directly between places or transitions. Places can contain zero or more tokens which indicate the state of the place. When all the incoming arcs of a transition are connected to places that each has at least one token, the transition may fire. A firing transition will remove one token from each place connected to every incoming arc and add one token to each place connected to every outgoing arc. Note that when a transition meets the conditions to fire, it does not necessarily have to. Nor can the order of firing be determined when multiple transitions meet this condition. As a result, the execution of Petri nets is non-deterministic.

Figure 3 shows an example of two traffic lights on an intersection using Petri nets. Places R1, G1 and O1 denote the three states of the first traffic light (i.e. red, green and orange). Similarly, the states of the second traffic light are represented with the places R2, G2 and O2. Place S is used to restrict the behavior of the traffic lights, in order to prevent both lights from being green or orange at the same time. Figure 3(a) shows the state of the traffic lights when they are both red. There are two transitions which meet the preconditions for firing and these are denoted by a red color. These transitions allow either traffic light to move from the red state to the green state. Figure 3(b) shows the state of the Petri net after the first traffic light has turned green. Note that the only valid transition in this state is for the first traffic light to turn orange, while the second traffic light is no longer capable of turning green.

Although Petri nets have made great contributions to the field of computer science, their application in other disciplines, such as workflow management, has only recently gained attention. With respect to business process modeling, Petri nets have been used to translate non-formal process models into formal process models for the purpose of analyzing or simulating these processes [9], [10].




(a) Both lights are red and are capable of transitioning to green

(b) The first light is green and the second light can no longer turn green

Figure 3 - Petri net of two traffic lights

### 2.2.3 Unified Modeling Language

The Unified Modeling Language (UML) is a general-purpose modeling language originally designed to be used in the field of object-oriented software engineering. UML is a standard which is actively managed by the Object Management Group (OMG). At the time of writing, the latest formally released version of UML is 2.3. The most recent version of UML specifies fourteen different diagrams, of which the Activity Diagram is the most used diagram for business process modeling.

An UML Activity Diagram consists of action nodes, object nodes and control nodes. An action node is the fundamental unit of behavior specification present in many UML diagrams. In the context of an activity diagram, an action represents some measurable piece of work which should be accomplished by a person or a computer. Object nodes represent the information which is consumed or produced by an action. From a functional perspective, an action is a transformation from a set of input objects to a set of output objects. Control nodes describe some aspect of the flow of control. Important control nodes are initial nodes, final nodes, decisions, forks and joins. The initial and final nodes represent the start and end of an activity diagram. Decisions are used to direct the control flow based on some information. Forks and joins can be used to split and merge the control flow in order to represent parallel processes [11].

There are two other important features of UML Activity Diagrams which are swimlane partitions and sub-activities. Swimlane partitions can be used to group actions on some common characteristic. Sub-activities can be used to aggregate an activity diagram into a single activity for use in other activity diagrams. Sub-activities facilitate composition and decomposition in activity diagrams.

Figure 4 shows an example of an UML activity diagram of an ordering process. The goal of the ordering process is to fulfill orders, ship products and perform billing tasks. A fork is used to represent that the billing tasks are ran in parallel with order fulfillment and shipping. A decision is used to represent that priority orders are shipped overnight, while all other orders are shipped using regular delivery. When the products have been shipped and the customer has paid the invoice, the order is closed and the process ends.

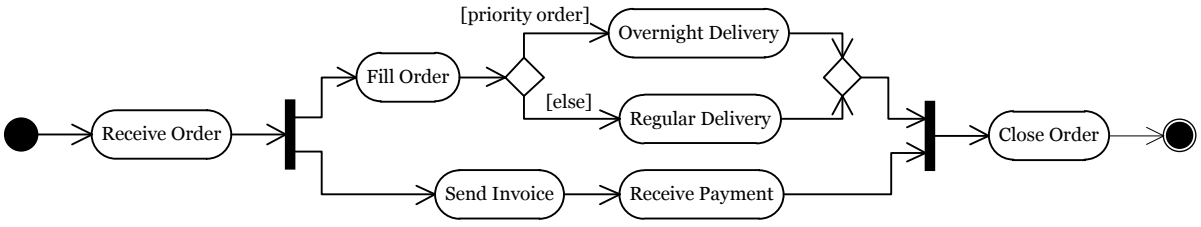



Figure 4 - UML activity diagram of an ordering process

Although the original purpose of UML was to assist the design of object-oriented software systems, more recent versions of UML have broadened its view to systems in general. Particularly, UML Activity Diagrams have been extended with several constructs that are attuned towards the organizational context [12]. While UML was never developed with the goal of business process modeling in mind, it has been used for this purpose extensively. The main motivators for process analysts to use UML are its great popularity, the large breadth of both methodological support and tooling support, the generic applicability of UML for conceptual modeling and UML's native support for extension with custom constructs. In other words, UML is an accepted and well documented general purpose modeling language.

### 2.2.4 Role Activity Diagrams

A Role Activity Diagram (RAD) is an element of the STRIM business process modeling methodology developed by Praxis Plc. for the elicitation, modeling and analysis of business processes [8]. As the name suggests, RAD are very similar to UML activity diagrams, with the main differences being that RAD emphasizes responsibilities, while UML activity diagrams emphasizes orchestration of the activities (e.g. aspects regarding the sequential or parallel execution of activities).

The primary constructs used in a RAD are roles, actions, interactions and decisions. Roles contain the actions and decisions that are performed by the man or machine with the assigned role. The interaction construct allows a role to communicate with another role, which also constitutes the only way how a relationship can be established between roles.

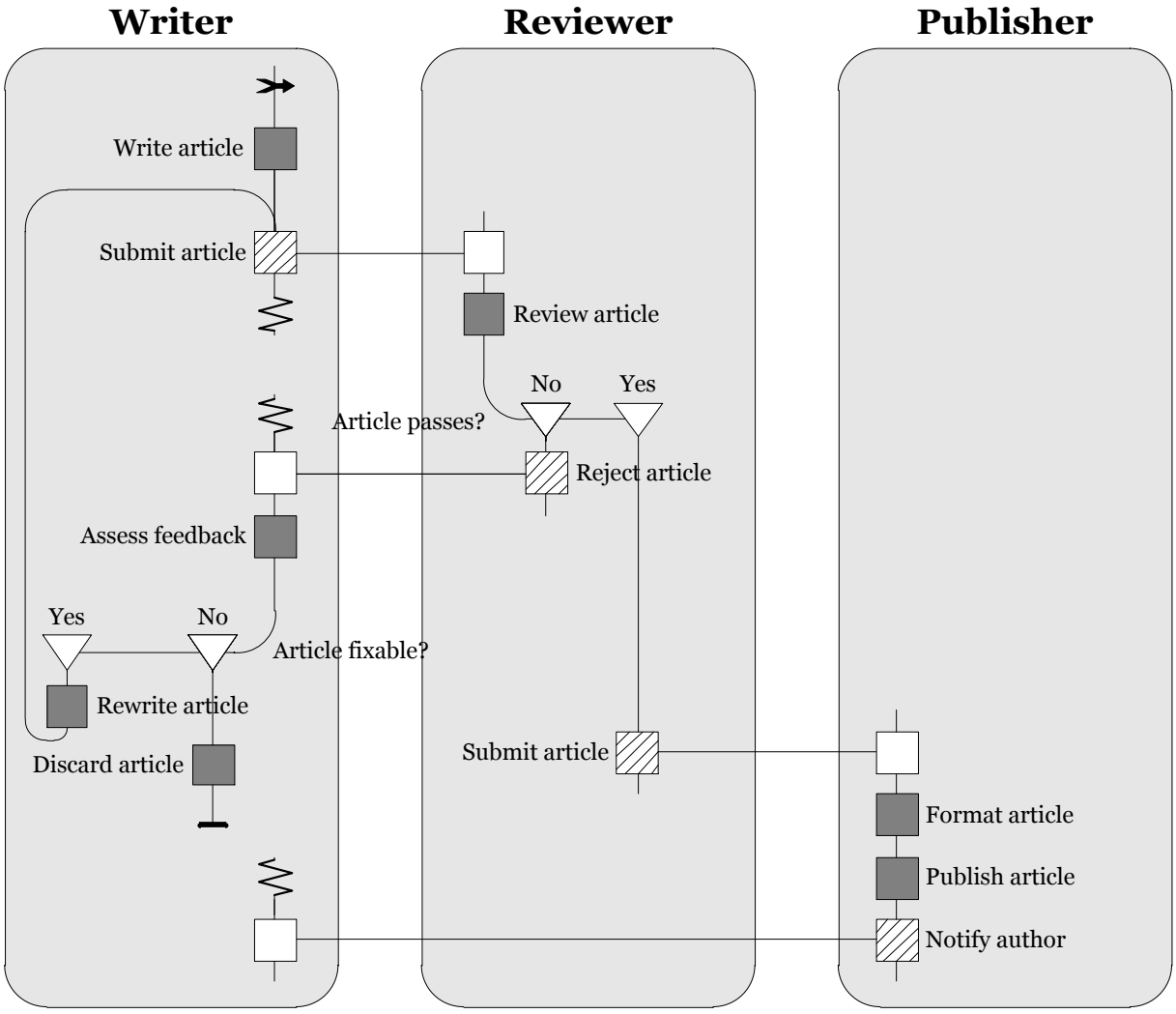



Figure 5 - RAD diagram of an article's lifecycle

Figure 5 shows an example of a Role Activity Diagram describing the end-to-end lifecycle of an article. The three large grey rectangles with the rounded corners represent the three roles involved with the lifecycle. The process starts with a writer who creates an article. When the article is deemed finished by the writer, it is submitted to a reviewer for approval.

The reviewer can either reject or approve the article. A rejected article is returned to the writer, who then uses the feedback to decide whether he wants to adapt the article or discard it. An adapted article enters the same review cycle as the written article, while a discarded article effectively ends the article's lifecycle and thus the process. An approved article is submitted to the publisher, who formats and publishes the article. The author of the article is notified of the article's publishing.

Note that there is an implicit timeline present in the diagram, as actions at the bottom are generally performed at a later point in time than actions at the top. Interactions are always denoted by horizontal lines as they represent synchronization moments where two or more roles come together and actively exchange information.

There are no formal semantics underlying a RAD as these models are aimed at facilitating shared understanding among stakeholders, rather than providing a basis for process simulation or execution. However, some research has aimed to formalize RADs by translating them to other representations such as Petri Nets in order to perform simulations [13], [14].

### 2.2.5 Event-driven Process Chains

An Event-driven Process Chain (EPC) is a business process modeling methodology aimed at creating business understandable models. Professor August-Wilhelm Scheer of the Saarland University developed the EPC method in 1992. Since its creation, the EPC method has grown to become one of the more popular business process modeling methodologies.

The core EPC constructs are events, functions and logical connectors. Events represent the pre- and post-conditions of a function, while a function represents an activity performed within the organization. Events are passive in the sense that they represent a state, whereas functions are active as they represent a transformation from one state to another. An EPC always starts and ends with an event. Both events and functions may only have one inbound and outbound relationship. Logical connectors are used to represent one-to-many relationships between events and functions.

An event cannot be preceded or succeeded by another event nor can a function be preceded or succeeded by another function. The latter constraint is generally ignored in practice as events between functions are often implicit.

Besides the core constructs, EPC also has several additional constructs which can be used to add additional information to the diagram. For instance, the organization unit construct can be used to represent which person or organization is responsible for the execution of a specific function.

Figure 6 shows an example of an EPC diagram of an ordering process, similar to the one shown in figure 4. Note that the biggest difference lies in the explication of the events. Events start and end the process. Moreover, all functions are the consequence of one or more events and also cause one or more events to occur. This approach towards modeling the ordering process creates a balanced view between what states the process can be in and what activities are executed.

The emphasis on logical connectors and functions makes EPC seem technically oriented, while it is in fact aimed at the business stakeholders, rather than the IT stakeholders. There are only little formal semantics underlying EPC, although some research has been performed on formalizing EPC [15].

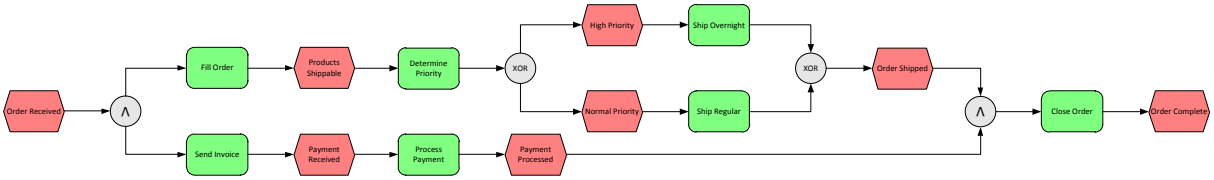



Figure 6 - EPC diagram of an ordering process

### 2.2.6 Integrated Definition for Functional Modeling

The Integrated Definition for Functional Modeling (IDEF) is a series of modeling languages originally designed to be used in the field of software engineering. The first IDEF modeling language came into existence in the mid-seventies as a byproduct of the Integrated Computer-Aided Manufacturing program of the United States Air Force. Since then, the IDEF family has grown to a set of sixteen modeling languages, each addition simply named IDEF0, IDEF1 and so on. However, only the first five IDEF languages have matured into well-accepted modeling languages, while the rest were never developed any further than their initial definition.

Of the sixteen modeling languages, IDEF0 and IDEF3 are the most suitable for business process modeling [16]. IDEF0 models business functions, while IDEF3 models business processes, so strictly speaking, IDEF3 is the only language of the IDEF family that is actually meant for business process modeling. In practice however, IDEF0 and IDEF3 are used to describe business processes, but from radically different viewpoints. Thus, the two languages should be considered complimentary rather than exclusive.

**IDEF0**

The IDEF0 modeling language has a limited syntax. In essence, everything is built upon the construct of a function. A function is an activity, similar to those used in flowcharts and EPC diagrams, which is represented by a rectangle. A function consumes input to produce some output. Additionally, a function is guided, regulated or constrained by controls and performed or executed by mechanisms. The output of one function can be the input, control or mechanism of another function. Inputs, controls, outputs and mechanisms (ICOM) are all represented by arrows which are connected to the function. Each ICOM arrow has a distinct side of the function rectangle to which it may connect. Input arrows connect at the left, control arrows connect at the top, output arrows connect at the right and mechanism arrows connect at the bottom [17].

Multiple functions in a diagram are ordered by dominance, with the most dominant function being placed at the top left of the diagram and the least dominant function being placed at the bottom right. A function dominates another function when it has more ICOM relationships leading to the other function than the other way around. This approach creates a cascading effect resulting in diagrams that have a waterfall like flow, which makes them easier to interpret.

Functional decomposition, on which we will further elaborate in 2.5.2, plays an instrumental role in IDEF0. All functions are always executed in a context, which is represented as the canvas on which the IDEF0 diagram is drawn. All inputs, controls and mechanisms which are not created by the functions within the diagram, originate from this context. Any function inside an IDEF0 diagram can be decomposed into a new diagram. In doing so, the ICOM of the function are mapped to the context of the new diagram. Numeric identifiers are used to cross-reference functions or ICOM arrows which appear in various diagrams.

Figure 7 shows an example of the order fulfillment of a furniture company. The process entails the creation of a design, the building of the furniture and the inspection of the built furniture. The ‘create design’ function transforms design ideas, the input, to actual designs of furniture, the output.

The designers are the mechanisms which execute the function, while the design process itself is controlled by the customer requirements. The designs which are the output of the design function are in turn used to control the building of the furniture. This shows that the output of a function does not have to be the input of another function, but can also serve as a control or mechanism.

From the perspective of functional decomposition, the diagram can be interpreted as a function called ‘Fulfill furniture order’, with the ICOM relationships I1, I2, C1, C2, M1 and O1. The parenthesis at the start of the control arrow called ‘Legal requirements’ indicate that the control does not appear in the parent diagram. This construction is called a ‘Tunnel in’. Likewise, the parenthesis at the end of the mechanism arrow called ‘Designers’ indicate that the mechanism does not appear in the decomposition of the ‘Create design’ function. This construction is called a ‘Tunnel out’.

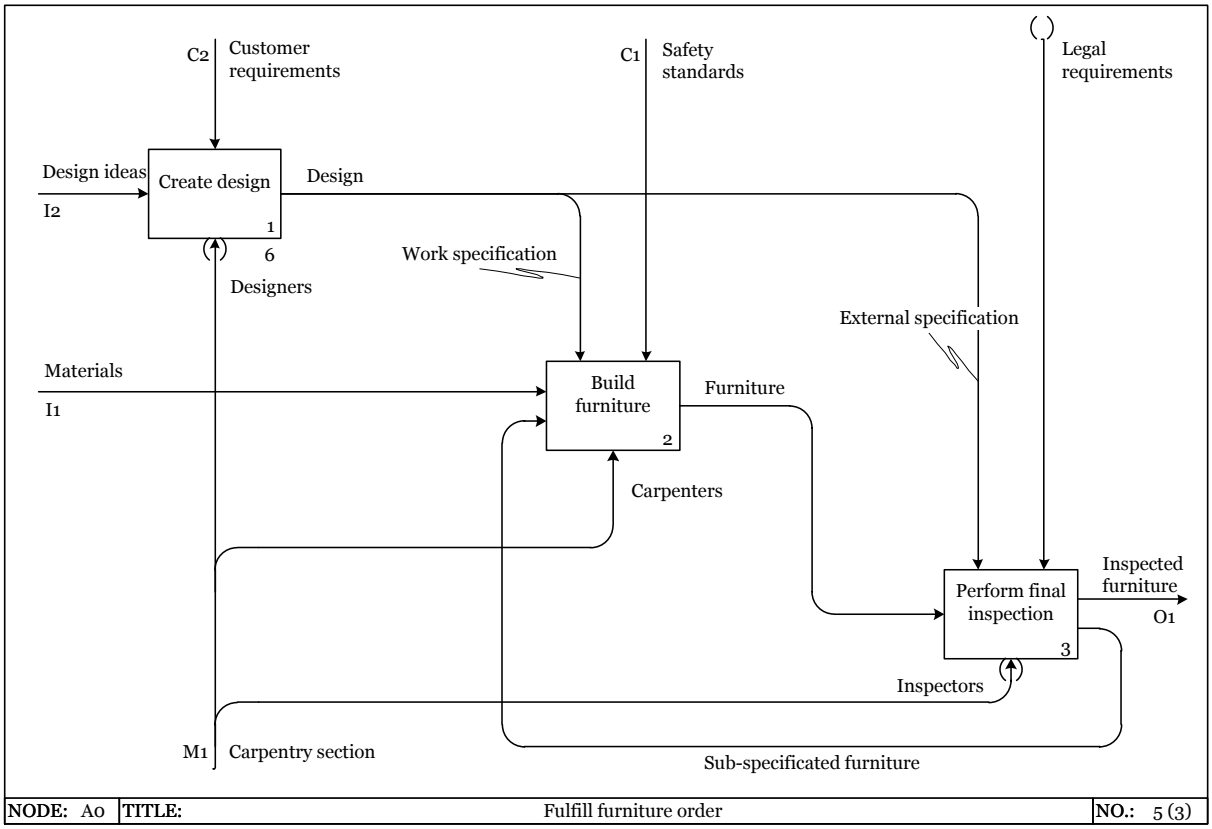



Figure 7 - IDEF0 diagram of a furniture construction process

The number below the 'Create design' function reveals that the function is decomposed in diagram number 6.

Note that unlike all other modeling languages, an IDEF0 diagram does not describe a sequence of activities. The relationships between the various functions indicate their dependencies, which may imply a causal relation, but this does not necessarily have to. The lack of temporal sequencing is one of the main advantages of IDEF0 as it allows users to model what the business does without forcing them to explicate how the business accomplishes this. Unfortunately, the tendency for people to interpret an IDEF0 diagram as a flowchart is also one of its disadvantages. Moreover, not everyone is comfortable with IDEF0 abstracting away time itself as it makes the language less intuitive. The lack of sequencing was one of the main motivators for the creation of IDEF3.

### IDEF3

IDEF3 focuses on the temporal aspect of business processes, which is in sharp contrast to IDEF0. Additionally, IDEF3 describes two different types of modeling languages. One for the purpose of describing process flows and one for the purpose of describing object state transitions. We will only cover the prior technique to limit our already lengthy description of IDEF.

Just like most other business process modeling languages, IDEF3 describes a business process as a series of activities. An activity is called a unit of behavior (UOB) and is represented by a rectangle. A causal relationship between two UOBs is called a precedence link and is represented by an arrow. Junctions can be used to split, join, branch and merge the control flow and are represented by small rectangles. The aforementioned constructs are the basic building blocks of IDEF3, which makes the language very similar to UML activity diagrams or EPC diagrams without the event construct.

Figure 8 shows an example of an IDEF3 process flow diagram of an ordering process. This example is based on the same example used in the UML activity diagram, shown in figure 4. Indeed, the diagram shows great similarities with the UML activity diagram and EPC diagram we covered.

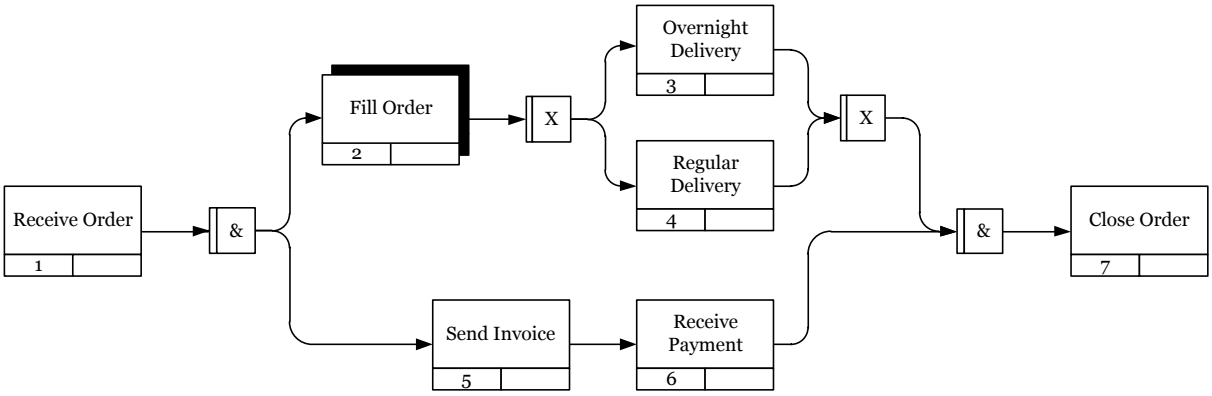



Figure 8 - IDEF3 diagram of an odering process

Note that there are no constructs which indicate the start or end of the process. The shadow below the 'Fill Order' UOB indicates that the UOB has been decomposed into another diagram.

IDEF0 and IDEF3 have some unique characteristics when compared to the other modeling languages. Both IDEF languages emphasize semantics over an elaborate syntax, making the IDEF languages more formal than all the other languages. Both IDEF languages support the use of functional decomposition and encourage its use. In fact, the title block constructs act as a drawing canvas which constrains the drawing space, thus forcing one to use functional decomposition. Another advantage of this approach is that IDEF diagrams work well on paper as the title block constructs are usually the size of a standard A4 paper. In contrast, other modeling languages may lead to the creation of very large diagrams, which introduce a range of problems.

Unfortunately, the emphasis on functional decomposition is also a disadvantage of the IDEF modeling languages. Large models may span across numerous diagrams, which makes it difficult to maintain overview of the overall process. Although there is always a root diagram at the highest level of the decomposition, the functions in this diagram do not reveal the depth of the decomposition. For this reason, IDEF models are often accompanied with a model tree diagram which describes the relationships between the various IDEF diagrams.

### 2.2.7 Business Process Modeling Notation

The Business Process Modeling Notation (BPMN) is a business process modeling methodology that aims to produce human understandable representations of business processes. The initial version of BPMN was developed by the Business Process Management Initiative (BPMI) in 2004. Two years later, the Object Modeling Group (OMG) adopted the language as a standard for business process modeling. As of 2011, BPMN is the most used notation for the modeling of business processes and considered the de facto standard [18]. At the time of writing, the latest version of BPMN is 2.0. Due to its immense popularity, we will use BPMN to provide examples of process modeling in the upcoming chapters.

BPMN is based on the same principles as flowcharts, but includes a much greater variety of constructs, making the language far more expressive than flowcharts. Besides flowcharts, the constructs present in BPMN show similarity to those used in UML activity diagrams and EPC diagrams. The core constructs of BPMN are events, activities, gateways and connections. Activities represent some kind of work which must be done, while events represent notable occurrences. Gateways can either be used to represent decisions which steer the control flow or can be used to split or merge the control flow (e.g. to perform activities in parallel). Finally, connections are used to establish the relationships between the aforementioned constructs. Each of these four constructs has a variety of more specific constructs which can be used to represent certain cases. For instance, the Timer Event is used to represent an event which occurs every so often.

Besides the core constructs, which are suitable for modeling most business processes, there is also a set of extended constructs. These constructs are meant for expert modelers who need to model

the more exotic cases. An example is the Event-Based Exclusive Gateway, which is a combination of a regular gateway, an Intermediate Event and a Start Multiple Event. One of the specific purposes of this complex construct is to represent scenarios in which various events can trigger the start of a process. Finally, BPMN supports swim lanes, which can be used to organize activities per role, and artifacts, which can be used to add additional information to the model, such as text annotations.

Figure 9 shows an example of a BPMN diagram of an ordering process. This example is based on the same example as the UML activity diagram shown in figure 4. The BPMN diagram shows great similarities to the previous diagrams which covered this example. Some additional details were incorporated in order in the example to showcase the expressive power of BPMN.

The circles represent events, which indicate notable occurrences during the ordering process. For instance, the ordering process is started when an order is received. The envelope inside the event element indicates that the event is triggered by receiving an actual message (e.g. a letter or e-mail). There is also an event present at the border of the 'Process Order' task. A boundary event represents a deviation or exception which can occur during the execution of the task. In this particular case, the event represents that the customer who placed the order is not known by the business. The symbol inside the event means that this is an Error Event and the bold face of the event means that the process ends when the event occurs. End Events with specific types can lead to follow-up actions, assuming the process in figure 9 is used as a sub-process in another diagram. Another use of events can be seen after the invoice has been sent. An Intermediate Message Event is used to represent the event of receiving payment, whereas an Intermediate Timer Event is used to represent a fourteen day delay. What this means is that when payment is not received within fourteen days, the order is escalated. Intermediate Events are events which do not start or end a process and are represented by a double border.

The diamonds represent gateways. The gateways with the plus symbol are Parallel Gateways while the gateways with the cross symbol are Exclusive Gateways. Their semantics are identical to the junctions of the IDEF3 diagram in figure 8. One interesting exception is the gateway after the 'Send Invoice' task, which is called an Exclusive Event-Based Gateway. This gateway is always followed by two or more Intermediate Events. The main purpose of this specific gateway is to indicate that one and only one event-based path succeeding the gateway may be taken. This is different from the Exclusive Data-Based Gateway which relies on data from the process instance to decide which branch to take. In contrast, the process instance is unaware whether the event 'Payment received' or '14 days passed' will happen first, only that when one event occurs, the process must continue on that path and ignore all other paths.

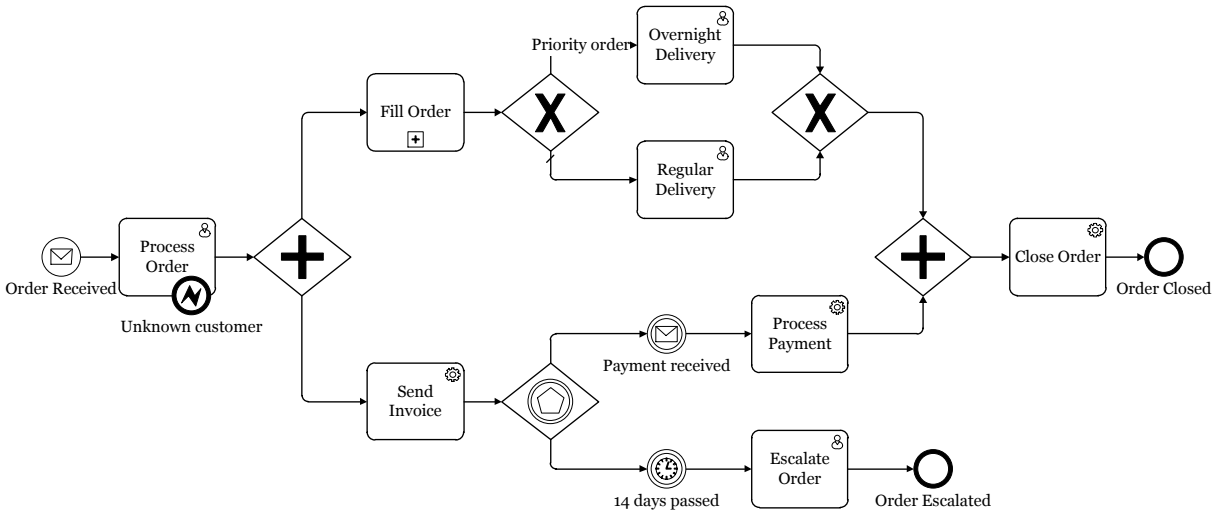



Figure 9 - BPMN diagram of an ordering process

The rectangles with rounded corners represent tasks and their semantics are similar to those found in the other examples. The symbol in the top right corner of the rectangle indicates the type of the task. A portrait specifies that a human is responsible for the execution of the task, while a gear specifies that a computer is responsible for the execution of the task. These help to differentiate between manual and automated tasks. The [+] symbol at the bottom center of the 'Fill Order' task points out that the task is actually a sub-process, meaning there is another BPMN diagram underlying it.

BPMN greatly emphasizes syntax over semantics. In fact, the set of available constructs in BPMN is so large that it is not always clear what construct is to be used in certain situations. In practice, over halve the users of BPMN only use the core set of constructs of BPMN [18].

### 2.3 *Business-IT alignment*

The state in which an organization is able to use information technology effectively to achieve business objectives is called business-IT alignment and has been a topic of great attention over the last few decennia [19]. Although both groups of stakeholders are nowadays aware of the underlying issues, achieving business-IT alignment remains an elusive objective [20]. One of the main inhibitors of business-IT alignment is miscommunication caused by the fact that business and IT possess little knowledge of each other's domain. The lack of mutual understanding is often paraphrased as business and IT speaking different languages or living in separate worlds. As a consequence of this misunderstanding, many software solutions developed by IT do not meet the demands of the business. Indeed, the effective translation of the demands of the business to a suitable software solution is often the most challenging part of an IT project [21].

BPM attempts to overcome this issue by letting the business express their demands in a process model, which in turn can be translated to a formal process model and executed by a process engine [22]. Instead of making IT responsible for determining the behavior of the system, which usually requires IT to understand the rules of the business, the business becomes responsible for the behavior. The role of IT is to create an adequate environment in which these process models can be created, analyzed, managed, stored and executed. This is often paraphrased as IT supporting the business, rather than taking it over.

We must add that we use the term business and IT to help facilitate our discussion, but realize that the distinction between the two is not as black and white as it used to be when the term was first coined back in 1995 [23]. As businesses rely more and more on IT to remain viable, the two groups have become intertwined and their boundaries have become blurred. A look at the current IT job market tells us, that a growing number of IT related vacancies no longer demands employees with a purely technical background, but rather seeks a mix of technical skills and so-called business skills. The latter refer to skills that include general understanding in the way businesses operate. Indeed, the demand for a Business-IT-hybrid employee is also recognizable in our current educational system, sprouting disciplines such as information sciences that educate its students in both fields.

### 2.4 *Language Theory*

Business process modeling incorporates various aspects from the theory of language. Firstly, we distinguish between natural language and artificial language as both types of language are used in process modeling. Secondly, two features of artificial languages, formality and notation, are explained.

#### 2.4.1 Natural language

A natural language is a language that has evolved as a means of communication among people and is the type of language we are most concerned with in our daily lives. Although natural language can be considered to be developed by us, we only have a limited understanding of the origin of natural

language, how natural language works and how we are capable of processing natural language. Natural language can manifest itself in several ways, namely: speech, gestures and writings.

### 2.4.2 Artificial language

An artificial language is a language explicitly designed with a specific purpose in mind. This is in sharp contrast to natural language, which serves a general purpose and has evolved naturally over time. There are various reasons for the creation of artificial languages, such as to ease communication or to serve as a basis for formal logic. Besides the purpose of an artificial language, there are two other aspects which are of interest to us:

- *Formality* refers to the precision of the syntax and semantics of the language. A sentence expressed in a fully formalized language, otherwise known as formal language, can be interpreted in one way and one way only. This property plays a crucial role in the field of formal logic, mathematics and computer programming. Languages with a formal syntax but informal semantics are called semi-formal languages.
- *Notation* refers to the type of visual representation of the language in terms of text and/or graphics. Programming languages are usually text-based, whereas languages for conceptual modeling are usually graphics-based. Contrary to what one might expect, a modeling language does not necessarily involve a diagrammatic notation, but for the purpose of this thesis when we use the term modeling language, we will be referring to a graphical modeling language [24].

## 2.5 *Conceptual Modeling*

A conceptual model is a descriptive model of a system. Although we do not have a clear definition of the term 'system', we will attempt to provide one based on its characteristics. A system is a collection of elements with a distinct boundary which separates it from its environment. A system has a purpose, is usually artificial and exhibits observable behavior. A system may consume some form of input and produce some form of output.

Given these characteristics, many things can be considered systems. Consider a vending machine as a system. The system has a specific purpose, to sell snacks and beverages to customers. Its input is money and one or more button presses, while its output is the release of a snack or beverage. The outer case of the machine provides a clear boundary in the physical realm.

Whenever we speak of modeling in this thesis, we mean conceptual modeling. This is a refinement over the term model, which generally speaking is any simplified representation of (a part of) reality. Such a definition is too broad to be of value for our research. Besides the topic of conceptual models and modeling languages, the act of modeling itself is also a topic of scientific attention [25]. We will elaborate on two principles from this field, namely that of ontological expressivity and functional decomposition.

### 2.5.1 Ontological Expressivity

The theory of ontological expressivity concerns the degree in which a modeling language can be used to describe the real world [26]. Some research has already applied this theory on business process modeling languages [27]. The real world can be described using ontology, the study of what exists in reality, with respect to its properties, its structure and how it relates to other parts of the real world. This assumes the thesis of *philosophical realism* which views reality as existing independently of its observers and being made up of structures and relationships. In other words, some elements in reality already form meaningful structures and we do not create these structures ourselves. This assumption makes it easier to map reality to a modeling language and back [28].

The degree of expressivity can be described using four types of ontological deficiencies of a modeling language shown in figure 10. These deficiencies are the four possible combinations over the constructs that either do or do not exist in the real world or the modeling language.

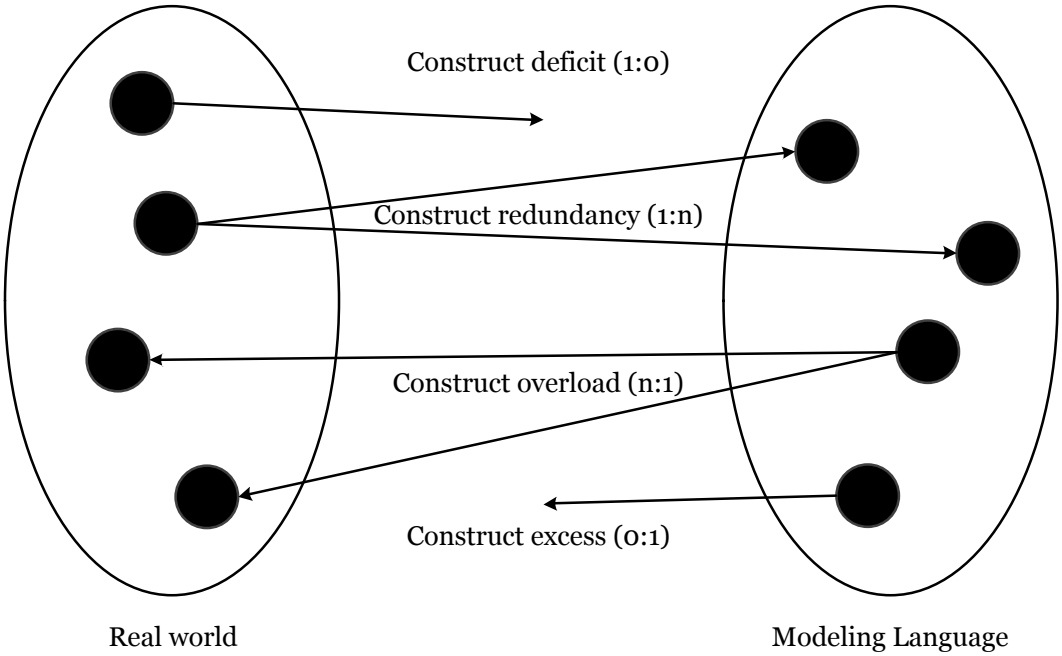



Figure 10 - Four ontological deficiencies of a modeling language

- *Construct deficit*. The modeling language lacks a construct which is present in the real world. The modeler will not be able to provide a direct representation of the real world construct and he or she will have to use one or more other modeling constructs to achieve the desired representation.
- *Construct redundancy*. The modeling language has two or more constructs which describe the same construct in the real world, thus creating an ambiguity in the modeling language. The modeler will encounter difficulty what modeling construct to choose for the description of the real world construct.
- *Construct overload*. The modeling language has a construct which describes two or more constructs in the real world. The modeler will encounter difficulty differentiating between the multiple real world constructs in his or her model.
- *Construct excess*. The modeling language has a construct which is not present in the real world. The modeler will not be able to use this construct in his model in a meaningful manner. Modeling languages that exhibit a high level of construct excess appear bloated and overly complicated.

### 2.5.2 Functional Decomposition

Functional decomposition is the act of dividing a system into smaller subsystems with the purpose of increasing understanding of the system. This goal is also one of the primary motivators for the creation of conceptual models [29]. Generally speaking, decomposition is a fundamental cognitive principle we humans use to understand the world around us. To quote Goguen and Varela (1979): 'The world does not present itself to us neatly divided into systems, subsystems, environments. These are divisions which we make ourselves.' We realize that this stance poses a conflict with the thesis of philosophical realism we mentioned in the previous subsection. A more appropriate thesis would be that of *critical realism*, which acknowledges that some of our interpretations accurately represent their real world counterparts, while other interpretations do not. Thus, critical realism can be seen as a relaxed form of philosophical realism with respect to its implications for the theory of ontology.

Reductionists argue that functional decomposition will ultimately lead to complete knowledge of the system, while holists disagree, arguing that "The whole is different from the sum of its parts". For instance, if one would take a car apart into its smallest constituent parts he or she might have gained full knowledge over the working of its combustion engine but not found an explanation why the steering wheel is on the left.

## 2.6 *Abstraction*

Abstraction has many different definitions and uses depending on its context. In its broadest sense, abstraction is a concept not associated with any specific instance. With respect to modeling, we define abstraction as the process by which an individual translates his observation of a part of the real world to a simplified form of knowledge. This definition exhibits great similarity to the act of modeling, which is on purpose as we believe that abstraction is a key component of any modeling activity.

We need to be able to abstract in order to reduce the infinite complexity of the real world to a limited set of concepts. This is a skill which we develop when we are young and keep on developing as our knowledge of the world expands. It is difficult to measure how well someone is able to perform abstract thinking much like it is difficult to measure how abstract a certain concept is. For instance, young children often struggle with the concept of time when learning how to read a clock or calendar. For adults, these matters seem quite concrete, while the effect of time dilation from the theory of relativity is perceived as abstract by many. When people are unable to give a concrete interpretation of an abstract concept, they will often have trouble understanding that concept.

In computer science, abstraction is the process by which programming constructs are defined with a specific meaning, while hiding away the implementation details [30]. The notion of hiding details plays an important role in any modeling activity as it allows one to leave out irrelevant information and focus on that which is deemed important. In business process modeling, information hiding is an important principle as these models tend to expand rapidly and a complete process description can contain an enormous amount of information.

## 2.7 *System Design Principles*

System design principle is an umbrella term we use to describe a principle which assists the design of a system. In this section, we use the same definition of a system as provided in section 2.5. We believe these principles can be used as generic guidelines for conceptualization. In other words, if conceptual modeling leads to the description of a system, then these principles can be used to guide the creation of this description. Strictly speaking, creating a model of a system is very different from designing a system. However as we already discussed in 2.1, modeling the business process guides the design of the business process and vice versa.

The principles we will explore are best known for their use in object oriented software design. Object orientation is a programming paradigm centered on the design of *objects*, which are data structures consisting of variables and methods. These objects can be considered as systems as they are created for a specific purpose, exhibit some behavior and have a well-defined boundary, often referred to as the scope of the object. The paradigm itself is not the focus of our attention. Instead, we focus on the design principles underlying object orientation as we believe that many of these principles transcend the computer science discipline. Note that there is considerable overlap between the various principles below.

The following principles will be covered: Modularity, Separation of Concerns, Single Responsibility Principle, Single Point of Entry, Ad-hoc Polymorphism and Information hiding.

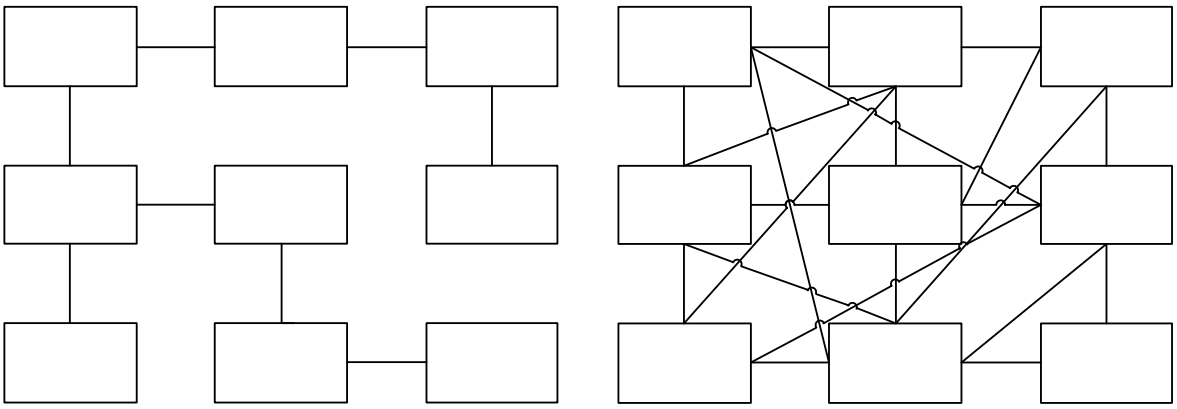
### 2.7.1 Modularity

Modularity concerns the design of a system as a set of separate, interchangeable components, called modules. Such an approach towards system design has many advantages such as improved coping with complexity, improved maintainability and reusability by enforcing logical boundaries between modules. The principle of modularity is supported by a variety of other principles and the next two subsections will describe the two most important factors driving modularity: the Separation of Concerns and the Single Responsibility Principle.

### 2.7.2 Separation of Concerns

Separation of Concerns is the principle whereby a system is decomposed into a series of subsystems that each addresses a specific concern. The identification and Separation of Concerns allows one to cope with the complexity of a large system, by splitting the system into several smaller systems. The Separation of Concerns principle is used in various disciplines, including architecture and information design [31]. Unfortunately, there is no generally accepted definition of what constitutes as a concern or how one should achieve such a separation. As a result, the principle seems superfluous: The idea that a complex problem is easier to solve when it is split into several simpler problems is rather evident.

The software quality metric called *coupling* is used to measure the level of dependencies a set of software modules have on each other. A set of modules that uphold the Separation of Concerns principle will have a low coupling, otherwise known as being loosely coupled, which is favorable over sets of modules that are highly coupled. The example in figure 11 shows two software systems consisting out of nine distinct modules. Figure 11(a) is loosely coupled as the nine modules have few dependencies on each other, while figure 11(b) is highly coupled as the nine modules have many dependencies on each other.



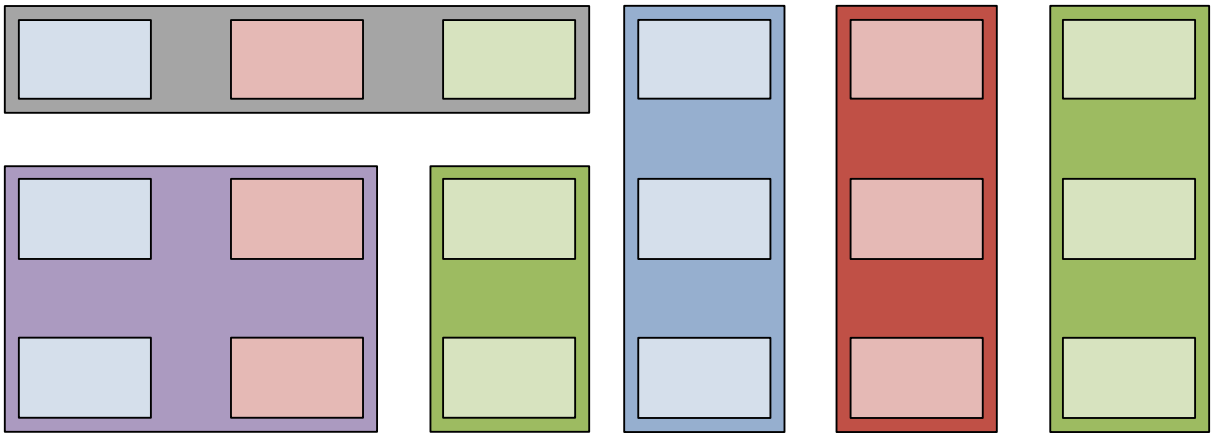
(a) Loosely coupled (b) Tightly coupled

Figure 11 - Two examples of coupling

### 2.7.3 Single Responsibility Principle

Single Responsibility Principle states that every system should have a single well-defined responsibility, which is yet another approach towards the modular design of a system. The importance of single responsibility is less evident during the initial design and implementation phase, but pays off when requirements change and existing systems have to be adapted. If a system has multiple responsibilities it is likely that these responsibilities have interdependencies, which means that a change to one responsibility might inadvertently have implications for the other responsibilities. In other words, systems with multiple responsibilities are more difficult to change without unexpected side effects. Unfortunately, just as the case with Separation of Concerns, there is no clear definition of what constitutes as a responsibility.

The software quality metric called *cohesion* is used to measure the degree of relatedness of a software module. Modules that uphold the Single Responsibility Principle will have a high cohesion, which is deemed favorable over modules with low cohesion. Figure 12 shows two examples of a software system consisting out of nine distinct functions (the inner rectangles). The color of these rectangles indicates some distinct responsibility of the function. These nine functions are grouped into three distinct modules (the outer rectangles).



(a) Low cohesion (b) High cohesion

Figure 12 - Two examples of cohesion

The two examples in figure 12 show two different ways of grouping the nine functions. The three modules in figure 12(b) are highly cohesive as their underlying functions share the same responsibility. The three modules in figure 12(a) show differing degrees of cohesion. The top module is the least cohesive as its underlying functions each have a distinct responsibility.

### 2.7.4 Single Point of Entry

Single Point of Entry is the principle whereby a system is accessible in one way and one way only. Although this principle is less common than the previous principles, upholding the Single Point of Entry principle can be advantageous in various specific cases. For instance, a building which has one entrance makes it easier to regulate access to, thus granting a security advantage. Moreover, visitors will not be burdened with having to figure out which entrance to use when there is only one entrance available. Similarly, large corporations often aggregate their numerous internal web applications and knowledge bases into a so-called enterprise portal, thus providing a unified and centralized place for all employees to fulfill their information demand.

### 2.7.5 Ad-hoc Polymorphism

Ad-hoc polymorphism is an object-oriented programming feature, which allows the definition of a method that is applicable to arguments of different types. A common example is the addition operator (+), which can be used on various types of arguments with different end results. For numbers, the sum is calculated while for text, the pieces of text are concatenated. Table 1 shows several examples of the addition operator being applied to arguments of different types.

Ad-hoc polymorphism can assist in the design of systems by helping to define more generic interfaces. Consider the example of a vending machine. One can choose to pay with cash or with a debit card. Although both payment methods are different types of input, they should result in the same behavior. Ad-hoc polymorphism helps to abstract from the specific payment methods and focus on the generic pattern of behavior. In other words, it assists in separating behavior specific to the type of input from the generic behavior which is applicable to all forms of input.

| Description | Function | Result |
|---|---|---|
| Calculating the sum of two integers | 2 + 3 | 5 |
| Calculating the sum of two floating point numbers | 1.5 + 2.35 | 3.85 |
| Concatenating two strings | “ab” + “cd” | “abcd” |
| Concatenating two sets | {1,2} + {3,4,5} | {1,2,3,4,5} |
| Adding one duration of time to another duration of time | 1h12m30 + 56m20s | 2h08m50s |

Table 1 - Various uses of the addition operator

### 2.7.6 Information hiding

Information hiding is a system design principle that allows one to protect the integrity of a system by exposing only those parts of the system which may interact with the environment. In software engineering, encapsulation is a programming language feature which can be used as an information hiding mechanism. Encapsulation is a well-known object orientation principle, mostly applied on classes and methods with the use of keywords such as *public* or *private*. These keywords regulate access to these classes or methods from other objects. When applied correctly, these objects are often referred to as *black boxes*.

Information hiding can assist the design of a system by separating the elements which are only of interest to the system from the elements which are of interest to the environment of the system. Consider the example of the vending machine. The environment should be able to buy products by interacting with the vending machine. Thus, the vending machine exposes several functions which allow the consumer to input his preference and transfer money. The consumer is not interested in how the vending machine is able to retrieve his product or is able to interpret cash. Additionally, the user should certainly not be able to influence these behaviors. For example, the consumer should be able to see the price of the products, but he should not be able to change the price of the products. From an object-oriented perspective, the variable that denotes the price of the product should be publicly accessible for reading, but only privately accessible for writing.

# 3 Fundamental topics

The previous chapter introduced the theoretical concepts which we will now use to discuss various fundamental topics. These topics will be used in chapter 5 to explain the various issues that are currently affecting business process modeling in practice.

## 3.1 *Natural language versus formal language*

Business process management places great emphasis on the use of models to describe business processes and these models must generally adhere to (semi)formal rules. One of the main inhibitors of successful business process management is the difficulty various stakeholders encounter in creating or interpreting such models [32]. At the same time, if one were to ask such a stakeholder to explain what the business process he is trying to model is about, he would have little difficulty coming up with an accurate description. How is it that we can more easily describe the real world using natural language than using a formal language?

### 3.1.1 Experience

For starters, we all have a tremendous amount of experience and knowledge of natural language. From the beginning of our childhood, we are taught to speak and read in one or more natural languages. Once we're capable of using natural language, we use it throughout the rest of our lives, increasing our experience and knowledge using natural language as the years pass by. In contrast, most formal languages are acquired after adolescence and are only used a fraction of the time we use natural languages.

That being said, basic artificial languages are easier to pick up than natural languages as they are considerably less complicated. For example, the decimal system describes ten unique digits and arithmetics, the most elementary branch of mathematics, describes four basic operators. Digits and arithmetic operators, combined with their semantics, allow one to perform basic calculations. The modern Latin alphabet contains 26 letters, but these letters are meaningless until they are combined to form words. Yet, even when letters form words, their meaning is still of varying (un)importance. Consider the following sentence: "*A vheclie epxledod at a plocie cehckipont near the UN haduqertares in Bagahdd on Mnoday kilinlg the bmober and an Irqai polcie offceir*". Although the letters of the individual words have been jumbled, one is still able to read the sentence with some increased effort [33]. The Oxford English Dictionary counts well over 250,000 words and the Kangxi dictionary contains close to 50,000 distinct Chinese characters. Although not all these words or characters are required to be proficient in their respective language, they do require considerably more effort to acquire and retain than artificial languages.

### 3.1.2 Syntax, Semantics & Pragmatics

The differences in syntax, semantics and pragmatics between natural and formal languages makes it more difficult to articulate thought in a formal language than in a natural language. Generally speaking, syntax describes the valid combinations of the elements of a language while semantics describes the meaning of these combinations. In natural language, these elements are words and the combinations of these words are sentences. In an artificial language, what these elements and their combinations are depends on the type of language. For instance, the elements used in first-order logic are the terms and formulas, while their combinations are called expressions. The elements used in a modeling language are the graphical constructs, while their combinations are their various uses in a diagram.

While semantics concerns the meaning of language, pragmatics concerns the intended meaning of language. This discrepancy in meaning is caused by the context in which the language is used. For example, the sentence "I am not angry" implies that the speaker is not angry. However, if the

sentence is uttered with a loud voice and the speaker's face is red, the intended meaning implies that the speaker is very much angry. This example shows that the context in which a language is used can lead to a discrepancy between semantic meaning and pragmatic meaning. While this discrepancy is part of the way we use natural language, it can lead to misunderstandings, which is a very undesirable property in certain fields (e.g. criminal law or mathematics). Formal languages, such as the mathematical notation, do not have this problem as they are completely independent of their context. As a result, the semantic meaning and pragmatic meaning of these languages are always aligned.

Modeling languages are also prone to the aforementioned discrepancy between semantics and pragmatics. For instance, BPMN prescribes a broad range of constructs one can use to draw a business process diagram, but does not include any rules on the positioning of these elements in the diagram. However in practice, people tend to layout these elements in a certain direction in order to incorporate a natural flow to the diagram. Semantically, the positioning of these elements has no meaning, but pragmatically, the positioning helps to place the element on a virtual timeline.

To conclude, pragmatics is a double-edged sword as it gives people more freedom to express themselves at the risk of creating misunderstanding. Formal languages do not suffer from this issue as they are context independent, but this also limits their freedom of expressiveness.

### 3.1.3 Purpose

Another factor which can hinder the effectiveness of a formal language is if it is used for the wrong purpose. A formal language is an artificial language which means that it was constructed with a specific purpose in mind. Problems arise when the purpose of a formal language and the purpose of its use in a specific situation are misaligned.

For instance, all Dutch bank accounts are identified by a unique 9-digit number. These account numbers are generally presented as three sequences of two digits and one single sequence of three digits, with each sequence being separated by a dot (e.g. 12.34.56.789). This presentation form can also be expressed using a *regular expression*, which is a specification written in a formal language with the purpose of matching text:

```
[0-9][0-9]\.[0-9][0-9]\.[0-9][0-9]\.[0-9][0-9][0-9]
```

However, validation of a Dutch bank account number also involves a mathematical equation called the 11-test, which is a test that checks if the weighted sum of all separate digits is a multiple of 11. Such a test can easily be expressed in various mathematical notations. Provided that $d$ is a nine-digit number and $d_i$ denotes the i$^{th}$ digit of $d$ then we could describe the weighted sum with the formula:

$$9 \times d_1 + 8 \times d_2 + 7 \times d_3 + 6 \times d_4 + 5 \times d_5 + 4 \times d_6 + 3 \times d_7 + 2 \times d_8 + 1 \times d_9$$

If the result of this formula can be divided by 11 without producing a remainder, the 11-test has held. We can generalize this formula to work for numbers with an arbitrary amount of digits and use a congruence relation with a modulus to represent that the result should be devisable by 11. In addition to the concepts we used in the previous formula, $\|d\|$ is the length of $d$ or in other words, the amount of digits present in $d$, and $x \equiv 0 \pmod{y}$ is a congruence relation which holds if $x/y$ has no remainder.

$$\sum_{i=1}^{\|d\|} (\|d\| - i + 1) \times d_i \equiv 0 \pmod{11}$$

The 11-test cannot be expressed using regular expressions as these are not designed to perform mathematical equations. Regular expressions work well for validating user input with a concise structure such as postal codes, phone numbers and e-mail addresses. The Dutch bank account example is quite straightforward in differentiating between the purposes of two formal languages and software developers would not generally try to perform math with a regular expression.

A more elusive example, which has been a subject of heated debate among software developers, is the suitability of regular expressions to process XML or HTML data. There are those who strongly oppose the practice as languages such as XML are context-free languages and regular expressions are designed to match regular languages [34]. For example, both XML and HTML support the nesting of tags to an arbitrary level of depth, which requires a recursive algorithm to parse the data. The use of recursion is not supported by regular expressions. However, regular expressions are often a more efficient solution than using a XML or HTML parser and there are many developers who advocate the use of regular expressions to extract information from XML or HTML data for this very reason.

These examples have shown that even while a formal language is designed for a specific purpose, this purpose might not be clear to its users and as a result, a formal language might be used to solve a problem it was not designed for. Moreover, there are situations where the suitability of a formal language to solve a problem is not clear cut and it becomes a matter of debate whether it is suitable or not. The purpose of a regular expression is to match text, but this does not make it appropriate for *all* pieces of text nor is it clear what properties a piece of text should have to warrant the use of regular expressions. The same can be said of business process models which aim to describe a business process as a sequence of activities. Such a description works well for some business processes while in other cases it does not work at all.

### 3.1.4 Expressiveness

Finally, even if a formal language is used for its intended purpose, it may still be difficult to use if it does not contain the right constructs to express whatever it is one is trying to represent. Once again, we turn to regular expressions for an example. Consider the validation of an identifier which must conform to the following constraints:

1. The identifier must only contain uppercase letters or digits.
2. The identifier must contain one and only one uppercase letter.
3. The identifier must be between 9 and 12 characters long.

The first two constraints can easily be captured by a regular expression such as

```
^[0-9]*[A-Z][0-9]*$
```

which can be read as zero or more digits followed by an uppercase letter followed by zero or more digits. However, the incorporation of the third constraint into this expression is not straightforward as it requires the two blocks that match the digits to be balanced, which is not a feature of regular expressions. Since it is not possible to directly express the intent of the third constraint, one has to overcome this limitation by using existing features to *mimic* this intent. An example of such a solution is to make all possible combinations explicit:

```
^(?:[A-Z]\d{8,11}  |\d[A-Z]\d{7,10}  |\d{2}[A-Z]\d{6,9}|\d{3}[A-Z]\d{5,8}|
  \d{4}[A-Z]\d{4,7}|\d{5}[A-Z]\d{3,6}|\d{6}[A-Z]\d{2,5}|\d{7}[A-Z]\d{1,4}|
  \d{8}[A-Z]\d{0,3}|\d{9}[A-Z]\d{0,2}|\d{10}[A-Z]\d?   |\d{11}[A-Z])$
```

This workaround has several obvious drawbacks. The length of the regular expression has sprung from 17 to 206 characters, which makes it much more difficult to interpret the meaning of the expression. Another drawback is that the maintainability of the expression has been severely reduced. If the length of the identifier is ever changed, this expression will require extensive editing to incorporate the change at the risk of creating inconsistencies in the expression. Some would argue that the creation of this regular expression should be automated to overcome these drawbacks.

The most sensible solution is to accept that it is unwise to place all three constraints in a single regular expression and to find an alternative solution. For instance, two simple regular expressions can be used to cover all three constraints. Alternately, if the regular expression is evaluated from a programming language, it is most likely a trivial task to incorporate the third constraint in that language.

Different business process modeling languages have different degrees of expressiveness. For instance, BPMN is far more expressive than regular flowcharts. Thus, someone who creates a flowchart might have to invest considerable effort to represent some piece of information which could be trivial to represent in BPMN. The theory of ontological expressiveness discussed in subsection 2.5.1 can be used to determine the expressiveness of business process modeling languages.

### 3.2 *Ambiguity in languages*

The previous section might have given the impression that natural language is superior to formal language. Indeed, the strength of descriptions in natural language is that everyone can more or less effortlessly understand them. Unfortunately, the weakness of these descriptions is that everyone understands them differently. The explication of knowledge to information and the interpretation of information to knowledge are accompanied by a varying degree of noise, as shown in figure 13. This noise is a result of the unique characteristics of the individual. We each have our own unique set of knowledge and life experiences, which shapes the way we create or interpret information. A person who reads a love story will interpret the story differently, depending if he or she has ever been in love.

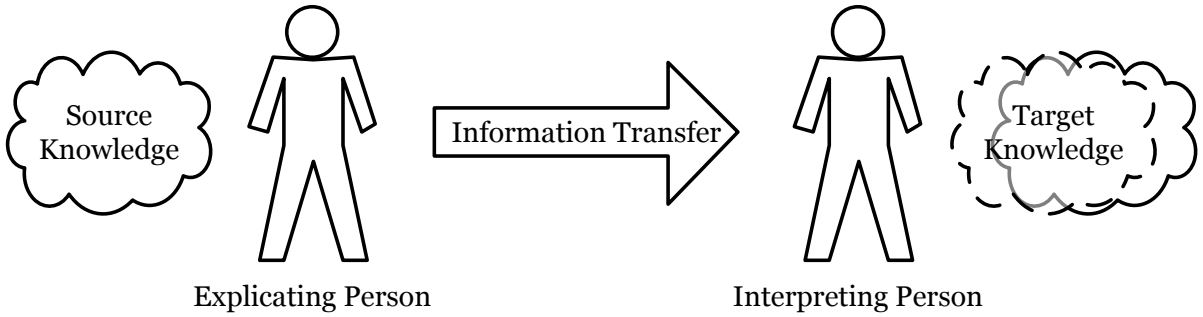



Figure 13 - Transferring knowledge using language

One of the primary causes of this interpretation bias in language is its inherent ambiguity. When we are aware of this ambiguity, we often employ corrective measures such as a follow-up question: 'What do you mean?' or 'Can you be more specific?'. However, when we are unaware of this ambiguity, we may interpret information and be convinced that we have understood its meaning, while in fact, the opposite is true.

Generally speaking, a piece of information is ambiguous if it can be interpreted in more than one way. Of course, given the fact that we all interpret information differently, one might wonder if it is possible to truly disambiguate information. We will assume a relaxed stance towards this philosophical question, by stating that even if information cannot be truly unambiguous, it can be unambiguous *enough*. In our opinion, information can be considered to be sufficiently disambiguated when it no longer leads to significant misunderstandings.

When we consider the ambiguity of a language, we can discern two types of ambiguity [35]:

- *Lexical ambiguity*, otherwise known as semantic ambiguity, means that a *word* has two or more meanings. For instance, the word *bank* has various meanings such as a financial establishment or the land alongside a river.
- *Structural ambiguity*, otherwise known as syntactical ambiguity, means that a *sentence* has two or more meanings. For instance, the sentence '*Dave took a picture of Jack with a camera*' can either mean that Dave used a camera to take a picture of Jack or that Jack was holding a camera while Dave took a picture of him.

Although ambiguity has its uses, it is generally perceived to be an undesirable property of languages as it can lead to misunderstanding. Politicians are keen in using ambiguous statements to avoid answering difficult questions or to avoid taking a specific stance on a subject. Ambiguity is specifically unfavorable during logical arguments as ambiguous facts can lead to the inference of incorrect conclusions. Formal logic alleviates this problem as it is based on a formal language and is thus completely unambiguous. In contrast to natural language, all sentences expressed in a formal language have one meaning and one meaning only.

However, even a seemingly formal language can contain ambiguity. A well-known example in computer science is the Dangling Else problem [36]. Consider the following piece of pseudo code:

```
IF x IF y ... ELSE ...
```

Without the use of parenthesis or some kind of ENDIF closure, it is unclear whether the ELSE statement belongs to the first or the second IF statement.

Considering that business process management involves the collaborative solving of a complex problem by various stakeholders, the presence of ambiguity is highly undesirable. As we mentioned in the introduction, one of the primary goals of BPM is to create a shared understanding of the way work gets done within an organization. Ambiguity directly impedes the achievement of such understanding, which is why semi-formal process models play such an important role in BPM. In fact, the elimination of ambiguity is mandatory when the goal of the BPM project is to create an enactable process model. Computers are by design unable to cope with ambiguity which means that for a computer to interpret a process model, it *must* be designed in a fully formal language.

## 3.3 *Abstraction*

The use of abstraction plays an important role in business process modeling. People from the field of IT are well versed in the use of abstraction. In fact, the evolution of computer hardware and software itself can be summarized as a growing stack of layers of abstraction [30]. It began with the creation of a logical gate which could perform some basic Boolean logic. Multiple logical gates were created and combined to perform simple tasks such as arithmetics and soon enough, the first calculator was created. The calculator allows one to perform arithmetics using the decimal system we're all used to while the actual calculation is performed using Boolean logic. However, the user of the calculator is not aware of this Boolean logic nor is he required to translate numbers to bits and back. In essence, the creators of the calculator have abstracted from the Boolean system present in electrical circuitry by adding an arithmetics layer. Contemporary computer software utilizes a large stack of layers of abstractions.

### 3.3.1 Leaky abstraction

Unfortunately, all forms of abstraction suffer from a phenomenon called *leaky abstraction*, which is a term coined by Joel Polsky in 2002 to describe the inherent imperfectness of all abstractions [37]. Consider the example of the calculator we used in the previous paragraph. The arithmetics layer is an abstraction built on top of the CPU which allows one to perform arithmetics using the decimal system rather than Boolean logic. In fact, the user does not require any knowledge of the inner workings of the calculator. All he expects is that if he hits the keys [5] [+] [3] [=] in order, the result [8] will appear on the display. However, at the layer of the CPU these numbers are sequences of bits and there is some hardwired limit to how many bits such as sequence may have. A very primitive 16-bit calculator might be unable to handle any value larger than $2^{16} = 65{,}536$. Clearly, characteristics present at the CPU layer have consequences at the calculation layer. This phenomenon is called leaky abstraction [38]. Someone who is unfamiliar with the characteristics of a binary systems will be unable to explain why he is not capable of calculating values larger than 65,536.

### 3.3.2 Conceptual abstraction vs. Scoping abstraction

Although the process of abstraction knows many forms, we made our own separation based on the manner in which it is applied in modeling:

- *Conceptual abstraction* is an information reduction method by which a concept is replaced by a more generic concept of a different type. For example, by describing a dog as an animal, one abstracts from the properties particular to a dog and leaves only those properties which are applicable to all animals. The concept of a dog has effectively been replaced by the concept of an animal. This type of abstraction is useful when one is less concerned with the specific details of a concept or wants to create multiple levels of abstractions that each describes a different type of information.
- *Scoping abstraction* is another information reduction method which is performed by replacing multiple concepts with a single concept of the same type. Considering the example of the dog, when there are many types of dogs it might be convenient to group these types into a new type of dog. In contrast to conceptual abstraction, no new concept is used. This type of abstraction is useful when creating overview is important. In conceptual modeling, the process of scoping abstraction is often referred to as *(de)composition*.

Distinguishing between conceptual abstraction and scoping abstraction will assist in the analysis of the application of business process modeling languages in our case study.

## 3.4 *Application of System Design Principles*

In this section, we will demonstrate the application of the various system design principles we discussed in section 2.7 on business process modeling. Not all principles are equally applicable and not all principles can be simultaneously applied as various principles conflict with one another. These conflicts are of particular interest as they require additional attention by those who are designing the business process. We will discuss these conflicts in section 3.5.

Some BPM research has already paid attention to these principles and we have observed the use of these principles in BPM projects, although often under the banner of software design principles such as object-orientation [39-41]. The use of system design principles on business process models is reasonable as software modules and business processes have several similarities. Business processes, as defined by business process models, are sets of activities with measurable triggers and results. With this definition in mind, the analogy with a method and its corresponding arguments and return-value is easily made.

The following principles will be applied to business process modeling: Modularity, Separation of Concerns, Single Responsibility Principle, Single Point of Entry, Ad-hoc Polymorphism and Information hiding.

### 3.4.1 Modularity

The idea of business processes as reusable and maintainable modules is a popular notion [42]. Consider any organization that has achieved success by performing some activity. Generally, they would want to take that activity and perform it elsewhere, preferably all over the world. In fact, the whole idea of *hard franchising*[2] revolves around organizations using an existing successful business model rather than developing their own.

Another advantage of modularity is maintainability, which is also applicable to business processes. For instance, if one department reorganizes and changes their way of doing work, it would be undesirable if other departments were forced to adapt as well. When a department is

[2] A soft franchise is the common type of franchise whereby a business adopts a certain brand but is free to manage their organization as they see fit. In contrast, a hard franchise not only includes a brand, but a way of work.

designed as a module (e.g. by specifying measurable incomes and outcomes of the department), changes within the department can be made without affecting other departments, assuming the department does not break the rules of the module.

Considering that many organizations change their way of work frequently, the creation of maintainable and reusable business processes seems desirable and modularity is a way to accomplish that. The following two subsections will discuss two system design principles which can assist in achieving modular business processes.

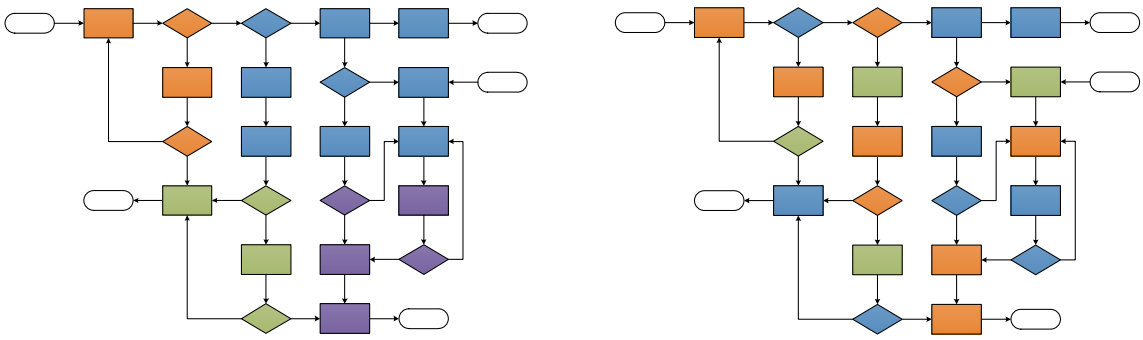
### 3.4.2 Separation of Concerns

Separation of Concerns is best known for its role in computer programming, but is applicable in a great variety of situations. *Edsger W. Dijkstra*, who coined the term, argued that Separation of Concerns transcends computer programming, or any design discipline for that matter, and stated that Separation of Concerns is a fundamental characteristic of all rational thinking. According to *Dijkstra*, Separation of Concerns is the *only* available technique one has to effectively order one's thoughts [43]. We support his statement and add that this principle is closely related to the use of decomposition. While decomposition allows one to divide a system in a set of subsystems, Separation of Concerns can be used to guide this division. In other words, combining decomposition with Separation of Concerns allows one to create a more meaningful division.

In business process modeling, Separation of Concerns can be challenging to achieve due to the great variety of concerns related to business processes [44]. Consider figure 14(a) which is a flowchart depicting a series of activities and decisions. The business process was designed with a certain concern in mind and the diagram reflects this concern through the use of color. The separation of the elements in the flowchart along these concerns is successful as the different colors depict closely related groups of activities and decisions. Furthermore, there are only a few relationships between the four groups implying they are loosely coupled which strengthen the modularity of the four groups. Figure 14(b) displays the same flowchart as figure 14(a) with the colors depicting a different concern. The four colors no longer depict elements that form a logical group and thus there appears to be no separation based on this concern.

The type of concern to separate on depends on the goals of the BPM project. Examples of such concerns are:

- *Responsibility*. A group of activities is performed by one person, function or organizational department. This is a common concern as most organizations are structured around functions and departments.
- *Accountable*. Similar to responsibility, a person or function is held accountable for the proper execution of a group of activities.



(a) Successful separation (b) Failed separation

Figure 14 - Example of Separation of Concerns in flowcharts

- *Result*. A group of activities produces a measurable and transferable result. Such a separation is of importance when knowledge transfer is important and communication between two parties is challenging, slow or expensive. For instance, when a helpdesk ticket is escalated to another department, the corresponding helpdesk report is transferred from one department to another. The report contains all the relevant information regarding the problem, thus reducing the need for employees from both departments to communicate.
- *Automation*. A group of activities is either performed by a person, a machine or a combination of both. Such a separation is often seen in BPM projects that aim to automate a considerable part of an existing business process.
- *Cost*. A group of activities has either low, medium or high costs. This separation seems less straightforward but has several widely applicable purposes. By placing the low-cost activities at the start of the process and pushing the high-cost activities to the end of the process, one can delay the cost of a process until it reaches the end. This is a highly desirable property of processes that do not necessarily run end-to-end. For instance, service helpdesks have an inexpensive employee at the start of the service process, which handles all the common cases. The more exotic issues are handled by experienced and expensive employees at the end of the process chain.
- *Location*. A group of activities is performed at a specific geographical location.
- *Time*. A group of activities is performed within a certain timeframe.

In practice, most of these concerns are related to some degree. Organizational departments usually have a defined set of responsibilities that produce some measurable result and reside on a single geographical location. The more these concerns are combined, the more well-defined the Separation of Concerns will be and the easier it will be to attach consequences to these concerns. Of course, this systematic approach towards the structuring of organizations is optimistic as in everyday businesses concepts such as responsibilities, departments and functions are quite fuzzy and their boundaries tend to be blurry rather than exact.

### 3.4.3 Single Responsibility Principle

As we have seen from our exploration of business process modeling languages in section 2.2, most business process modeling languages are based on the notion of an activity: a piece of work performed by an employee or computer which in some way contributes to the execution of the business process. Single Responsibility Principle dictates that every system should have a single well-defined responsibility. When applied to activities, we get a very meaningful and powerful principle: Every activity should have a single well-defined responsibility.

This principle is already incorporated in various business process modeling guidelines, though it is not called by this name. For instance, the OTOPOP principle of the Common Reference Architecture (CORA) model, states that each Use Case should be performed by one person at one place at one time. Although they apply this principle to Use Cases, it is part of their methodology to map Use Cases to activities within business processes [45]. On a related note, various research attempts have aimed to extract software requirements such as Use Cases from business process models [5], [46], [47].

Consider the example of the ordering process described in figure 9. The process includes a task ‘Send Invoice’. However, sending an invoice is no trivial matter as an invoice needs to be created and reviewed first. These activities are performed by separate individuals, perhaps even in separate departments. Thus, the task is converted into a sub-process, which is described in figure 15. In this diagram, the first task ‘Create Invoice’ is yet again decomposed into another sub-process, which is described in figure 16. This manner of decomposition could continue until the sub-process describes actual actions such as ‘Pick up a pen’ or ‘Hit Enter’, but such a level of granularity is pointless. In fact, the tasks described in figure 16 are usually found in a work instruction manual, rather than a business process model.

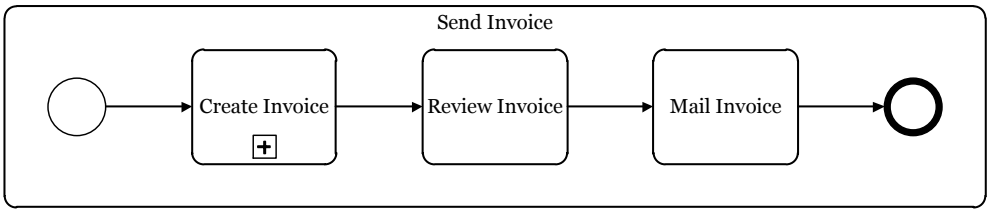



Figure 15 - BPMN diagram of the 'Send Invoice' sub-process

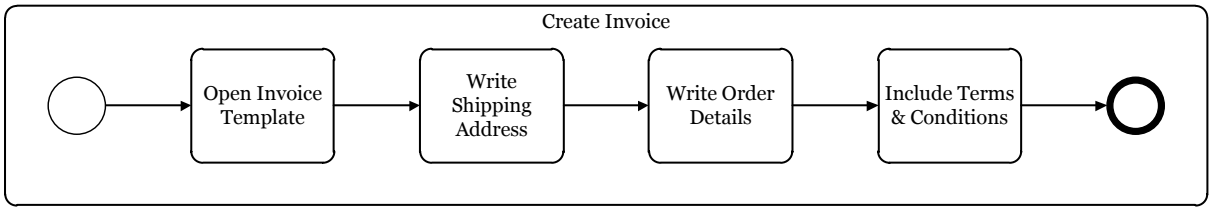



Figure 16 - BPMN diagram of the 'Create Invoice' sub-process

### 3.4.4 Single Point of Entry

The Single Point of Entry principle can be useful in the decomposition of business process models. To illustrate its importance, once again consider the decomposition of the first activity of the diagram in figure 15 into the diagram in figure 16. The use of decomposition works well in this example, as the sub-process 'Create Invoice' has one input and one output, that correspond with the single start and end event present in its underlying process diagram.

However, complications would arise if the process depicted in figure 16 would have multiple start events as these would need to be mapped to an equal number of incoming process flows in the parent diagram. This would in turn require some way to differentiate between various process flows in order to represent which process flow connects to what start event. Such a construction is not possible in BPMN.

To avoid these problems, the BPMN specification states that the business process underlying a sub-process *must* have one and only one start event. The same rule does not apply to end events, although it is strongly recommended that any additional end events be of a specific type such as a message, escalation or error event and assigned a distinct label. The parent diagram can then represent the consequences of these end events through the use of boundary events.

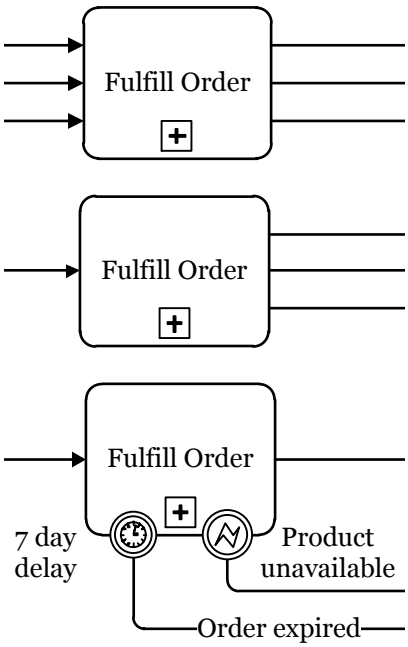



Figure 17 - Three uses of sub-processes in BPMN

To illustrate, Figure 17 shows three different uses of a sub-process. The top two examples are a result of attempting to create a sub-process from a tightly coupled set of activities. The bottom example is the most meaningful as the sub-process has a single start and end point. Deviations from the main process flow are described through the use of Intermediate Boundary Events.

To conclude, the more process flows cross a sub-process boundary, the less meaningful the decomposition is and the harder it is to interpret its use in diagrams. To avoid these problems, uphold the Single Point of Entry principle when defining sub-processes.

### 3.4.5 Ad-hoc Polymorphism

While ad-hoc polymorphism is a programming feature rather than a design principle, it can have a particular but useful purpose in business process modeling. A common pattern in business process modeling is that a business process can be started in varying ways.

Consider the case of a customer who wants to file a complaint as shown in figure 18. From the perspective of the business, the customer complaint process can either be initiated by a complaint via e-mail, telephone or letter. In logistics, this is also known as multi channeling. Usually, each specific channel has its own way to be handled but at some point the manner in which the complaint is processed is no longer specific to the way the process was started.

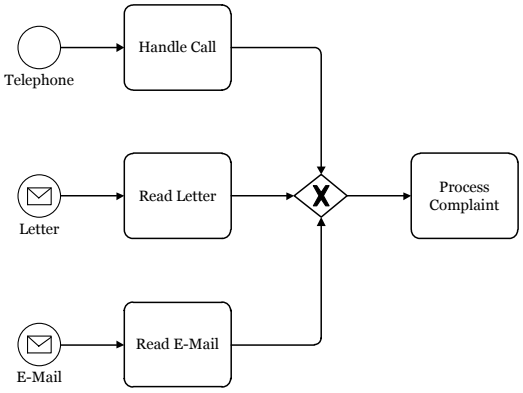



Figure 18 - BPMN diagram of a customer complaint process.

From a process modeling perspective, if one would create a separate process for each channel, one would observe considerable overlap between these processes. This is a less than optimal solution, because a change in the overlapping part would require a change in all processes. To limit this overlap, one could design a single process that incorporates each channel up until the point where the process becomes generic for all channels. From a process modeling perspective, this would create a process with multiple starting paths which at some point converge and continue on a single path, just like the BPMN diagram in figure 18 shows.

### 3.4.6 Information hiding

In business process modeling, information hiding does not seem straightforward to apply. However, we argue that information hiding could play a significant role, namely to hide technical details (i.e. that information which is only relevant for the formal correctness or enactment of the process model)

One of the three goals of business process management is the creation of executable process models with the purpose of providing direct technical support to the business process. This requires a computer to be able to interpret a process model in such a manner that the process it describes can be executed by the computer. In turn, this requirement means that the process model must be described in a formal language as this is the only type of language a computer is capable of processing. Unfortunately, the transformation of a process model which is understandable for humans into a process model which is understandable for machines involves the addition of a large amount of technical details. These details are meaningful for IT stakeholders who understand their intent, but confuse other stakeholders. We argue that for all information in a process model, there should be a clear distinction what information concerns the technical implementation and what information concerns the actual business logic. Non-IT stakeholders are almost always concerned with the latter information.

Although numerous BPM efforts attempt to avoid incorporating technical details in their business process models, there are nearly always choices at the technical level that have consequences for the process model [48]. For instance, in many BPM projects, BPMN is used to describe a business friendly process model while the technical process model is implemented using BPEL. Of course, such a separation introduces a variety of conceptual mapping issues between the two models. Indeed, much research has been performed of the mapping of BPMN to BPEL and back [49], [50]. At the moment, BPEL appears to be replaced by BPMN 2.0, which is the newest version of BPMN and allows for both the description of process models and their enactment.

## 3.5 *Conflicting principles*

All principles we have covered can be applied to business process modeling in some desirable way. However, some principles inhibit the effectiveness of other principles, which requires the designer of the model to make a tradeoff between these principles.

### 3.5.1 Separation of Concerns versus Single Responsibility Principle

Applying both Separation of Concerns and Single Responsibility Principle implies the creation of a set of activities that each have a distinct purpose which is not present in the other sets of activities. Although this seems to be highly desirable, in most cases it also highly infeasible.

Consider the following computer programming related example. A content management system allows logged in users with the right credentials to perform a variety of tasks such as publishing articles. The function that is responsible for checking if a user is logged in can either be placed in the module that is responsible for publishing articles or it can be placed in a separate module that is responsible for authentication and authorization. In the prior case, Separation of Concerns is successful as the article publishing module is able to perform its task of publishing articles without any dependencies. However, single responsibility is lost as the module is responsible both for publishing an article and checking user credentials. In the latter case, the Single Responsibility Principle holds as there are two modules that each have their own distinct purpose. Unfortunately, Separation of Concerns is lost as the article publishing module relies on another module to perform its task. These tradeoffs occur in the most simple of programs and it is up to the programmer to make these choices as there is often no right answer.

Separation of Concerns results in sets of systems that share as little concerns as possible while the Single Responsibility Principle results in a set of systems that each has a single distinct purpose. Maintaining both principles becomes harder as the complexity of systems increases. At some point the designer will encounter a responsibility that does not belong anywhere. These arbitrary or miscellaneous responsibilities are then often grouped in generic systems. For instance, many programmers often place arbitrary functions in a *Utility* class.

In business process modeling, these tradeoffs can occur as well. For example, emphasizing Single Responsibility Principle can lead to process models whereby each activity is executed by a distinct employee, at the cost of having more dependencies between employees. This is comparable to Frederick Winslow Taylor's scientific management and his views on manual labor. Taylor argued that a complex task such as the manufacturing of an automobile should be reduced to a series of atomic tasks that each is executed by a distinct employee [51]. Although this radically increased the efficiency of car factories, the high level of interdependencies also means that if an employee is no longer available, a certain task in the process chain can no longer be executed, subsequently halting the global process.

In contrast, emphasizing Separation of Concerns can lead to process models that are completely independent of each other. To relate to the previous example of the car factory: All laborers are capable of building an entire car by themselves. Conceptually, each laborer can be viewed as a small car factory. Failure of one employee may lower the throughput of the overall factory, but will not completely halt its process. However, the costs of replacing a laborer are extremely high as each new laborer must be taught how to build an entire car.

To conclude, the designer should make his own tradeoff between Separation of Concerns and Single Responsibility Principle, depending on the nature of the business process he is trying to model. It is not possible to fully support both principles nor is it wise to push either principle to its extreme.

### 3.5.2 Single Point of Entry versus Ad-hoc Polymorphism

The principles of Single Point of Entry and Ad-hoc Polymorphism are in direct conflict with one another. The prior emphasizes the importance of creating a single connection between the system and its environment, while the latter emphasizes the importance of creating multiple connections.

Consider the example of a customer complaint process from subsection 3.4.5. Applying Single Point of Entry to this process means that the process can only be invoked in one way. In terms of a BPMN diagram, the diagram must have one and only one Start Event. Now say we want a customer to be able to file a complaint either by e-mail or by telephone. Upholding Single Point of Entry means the creation of a second customer complaint process, while Ad-hoc Polymorphism would allow the first customer complaint process to be adapted in order to allow both means of invocation. Choosing between both options depends on the degree of overlap between both processes. If the helpdesk is capable of processing both mail and answering telephone calls, Ad-hoc Polymorphism is probably the best choice. However, it might preferable to create two helpdesks, perhaps even in distinct locations (e.g. hiring a call center in India), in which case it is better to keep both processes separate.

To conclude, the degree of overlap between two processes can guide the decision whether to separate or merge two processes. Figure 19 shows an example of two overlapping business processes in a BPMN diagram. The first two activities of both processes are unique, while the last two are the same. The greater overlap, the more incentive there is to merge the two processes.

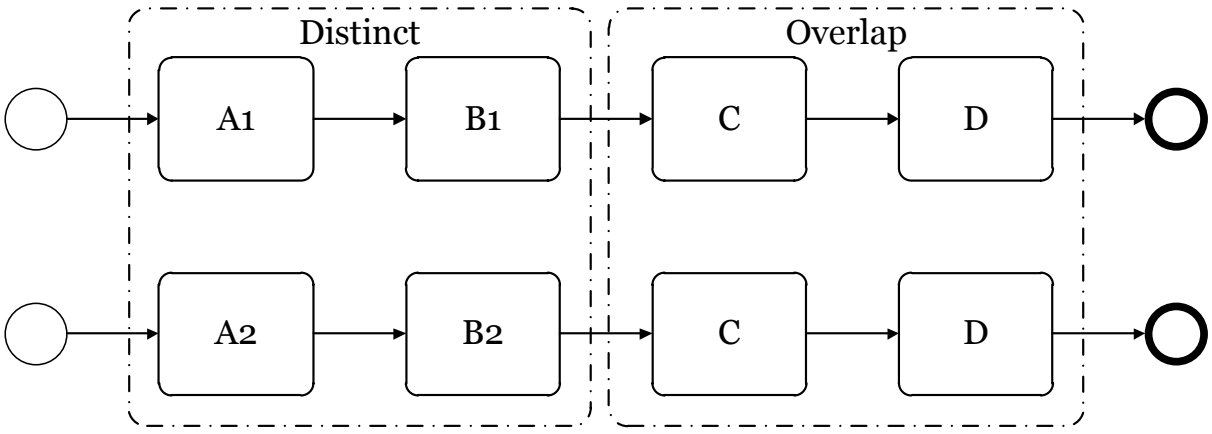



Figure 19 - Overlapping business processes in a BPMN diagram

# 4 Case study

We performed a case study of a project involving the elicitation and design of business processes. Although the project was not explicitly called a BPM project by its stakeholders, it did exhibit several characteristics of a BPM project which were of interest to us, specifically:

- The goal of the project was to design a new way of work, which falls in line with one of the motivators for engaging in BPM.
- The way of work was to be described using various notations, including flowcharts and BPMN, with the goal of creating a shared understanding among all stakeholders. This goal is identical to one of the primary goals of business process modeling.
- There were issues regarding the created diagrams. As our research aims to uncover such problems, these were of great interest to our research.
- Both business type and IT stakeholders were involved in the project.

This chapter is structured in three parts. First, an outline of the project is provided in order to set the context of the project: what stakeholders were involved and what they were trying to achieve. Secondly, an analysis of the various business process models used in the project is given. Thirdly, the results of the analysis are summarized to illustrate which issues related to the modeling of business processes played a part in the project.

## 4.1 *Outline*

The case focuses on a project for the government of a large municipality and was centered on the redesign of existing business processes. These processes described services provided by the municipality to its citizens. The following three sub chapters give a chronological description of how the project progressed. Each chapter focuses on a specific stakeholder, but does so without breaking the overall chronological order.

All information presented here was gathered through the architect stakeholder, which might pose a somewhat biased view on the project. The outline of the case and its subsequent analysis has been performed objectively relying on factual data as much as possible. We do not believe that neither the architect's interpretation nor our own has introduced any significant distortion in our analysis.

### 4.1.1 The Business

The initial project focused on the redesign of the business processes of one specific organizational unit. A consultancy firm was contracted to analyze the existing process and to design a new and improved version. The redesign resulted in a 151 page large document describing the process as a collection of several sub processes. Each sub process was modeled using a basic flowchart like notation and each step in the process was then further elaborated using a table template description. The document was accepted by the various stakeholders, but as the project moved onward, some concerns were raised about the apparent lack of overview. Although the document accurately described the various sub processes concerned with the execution of the overall process, it was unclear how these sub processes related to each other and there was no clear view of the overall process.

To alleviate these concerns, a large poster was created in collaboration with various stakeholders using several workshops that incorporated all the sub processes in one large model. This unified process model, also known as the *large poster*, counted well over two hundred elements, but was considered to provide more of an overview than the original document and was thus accepted by the various stakeholders. More importantly, the large poster allowed one to easily oversee the relationships between the various processes.

### 4.1.2 The Architect

Several months later, an architect of a medium-large software development company was contracted to create an architectural description of the necessary business, information and technical changes. When the architect attempted to grasp what the project was about, he inevitably ended up with the large poster that had been created and he raised some concerns about its content. He was mostly concerned with the level of detail as both the documents and the large poster were a low-level description of the process. According to the architect, the process designs lacked a proper high-level description, which concerned the process as a whole. He argued that it would be beneficial to all parties involved if such a high-level description were to be created. Furthermore, the architect noted that there was no clear definition of some of the notational constructs used in the diagram. In other words, the diagram lacked semantics.

Stakeholders of the large municipality were uneasy to accept these concerns, as they held their process designs, such as the large poster, in high regard. The architect understood that it would be difficult to get these stakeholders to create a new model, let alone use a different notation or tool. However, two new versions of the large poster were created, one using a higher level of description with all the low level details hidden and one incorporating both a high and low level description. These new models were quite similar to the original model in terms of notation and style, but at a conceptual level the new models described a different type of information. Although the stakeholders of the large municipality, albeit reluctantly, accepted the new model, the architect noticed that in everyday use, most stakeholders still turned to the low-level model for guidance as it contained the most detailed description.

### 4.1.3 The Technicians

In a follow-up project, a small software development company was contracted to facilitate the development of the information systems which would support the designed business process. Despite the fact that it was clear from the very beginning that an information system would need to be developed to support the business process, the involvement of an actual IT party was deliberately delayed until the design of the business process was finished. One of the requirements of the stakeholders of the municipality was that the business process should be independent of any technical solution. They were concerned that the early involvement of an IT party would jeopardize this requirement.

Requirement analysts of the software development company attempted to use the various documents and models as a basis for requirements elicitation, but encountered difficulties in doing so. The various versions of the large poster were considered to be overly complicated and lacked the information they really needed, such as the input and output of the various activities. As a result, the requirements analysts of the software development company decided to create a new model to better fit their needs. Unlike the architect, they decided to go with a different modeling methodology and used BPMN in combination with a professional process modeling tool. Even though the new model was in many ways an improvement over the original designs, it was not accepted by the stakeholders of the municipality as they no longer recognized their business processes and argued that the new model lacked the overview the original model had. This puzzled the requirements analysts as they believed that the new model provided a much better overview of the process through the use of decomposition.

Stakeholders of the municipality also had difficulties understanding the model as they were unfamiliar with the notation of BPMN. They explained that the use of a business process modeling methodology such as BPMN was discussed during an earlier phase of the project, but was explicitly avoided in favor of a light-weight notation that was easier to pick up.

## 4.2 *Model analysis*

An analysis of the aforementioned models was performed with the purpose of assessing what information the models represented and how well they represented this information. Several concepts will be used in our analysis which will now be defined:

- A *model* is a representation of a (part of) reality. Although we will mostly focus on graphical representations, a model can consist out of any combination of representations. In the case's project, the graphical representations were often accompanied with documents to provide a more in-depth description. We state that the actual model is the combination of the graphical representation with the accompanied documents.
- A *view* is a single representation of the information contained within a model. The information represented in the view is a subset of the information present in the model. Views allow one to look at a model from various perspectives, often also referred to as viewpoints. Sometimes, it is necessary to create views because the model is too complex to be represented in a single view. In other cases, there is only one view in which case one could consider the view and the model to be same. Generally speaking, one creates different views on models to satisfy specific needs for information of different stakeholders.
- A *diagram* is a particular instance of a view which uses a graphical representation (e.g. flowcharts).
- An *element* is a symbol or object used in the diagram (e.g. rectangles, diamonds or arrows).
- A *construct* is a type of element combined with its semantics. For instance, the flowcharting technique prescribes an activity-construct and a decision-construct. Each construct has a specific meaning and has its own distinct graphical representation. For instance, the activity-construct represents some kind of task that should be performed and is presented by a rectangle. When related to our definition of elements: Constructs are types and elements are instances of these types.

### 4.2.1 Large Poster – First Version

The first object of our analysis was the initial version of the large poster shown in figure 20. The poster was created by the stakeholders of the large municipality and represented one of the first modeling efforts in the project. Due to the considerable amount of time and money spent on the creation of the model, the end result was held in high regard and served as a means to create a shared understanding among all the stakeholders.

The diagram was designed with a basic diagramming program using the notation of a flowchart. Besides the usual constructs found in a flowchart such as activities and decisions, the diagram also describes the construct of a *global process step*. This is a collection of activities and decisions that, together, indicate a significant piece of work generally performed by a single employee. A global process step produces a measurable result which is transferrable to another employee and thus serves as an input for other global process steps. The most notable impact of these global process steps on the diagram is the abundance of colors used to describe these global process steps. Each global process step has its own distinct color, which can also be found in other diagrams and documents created throughout the project.

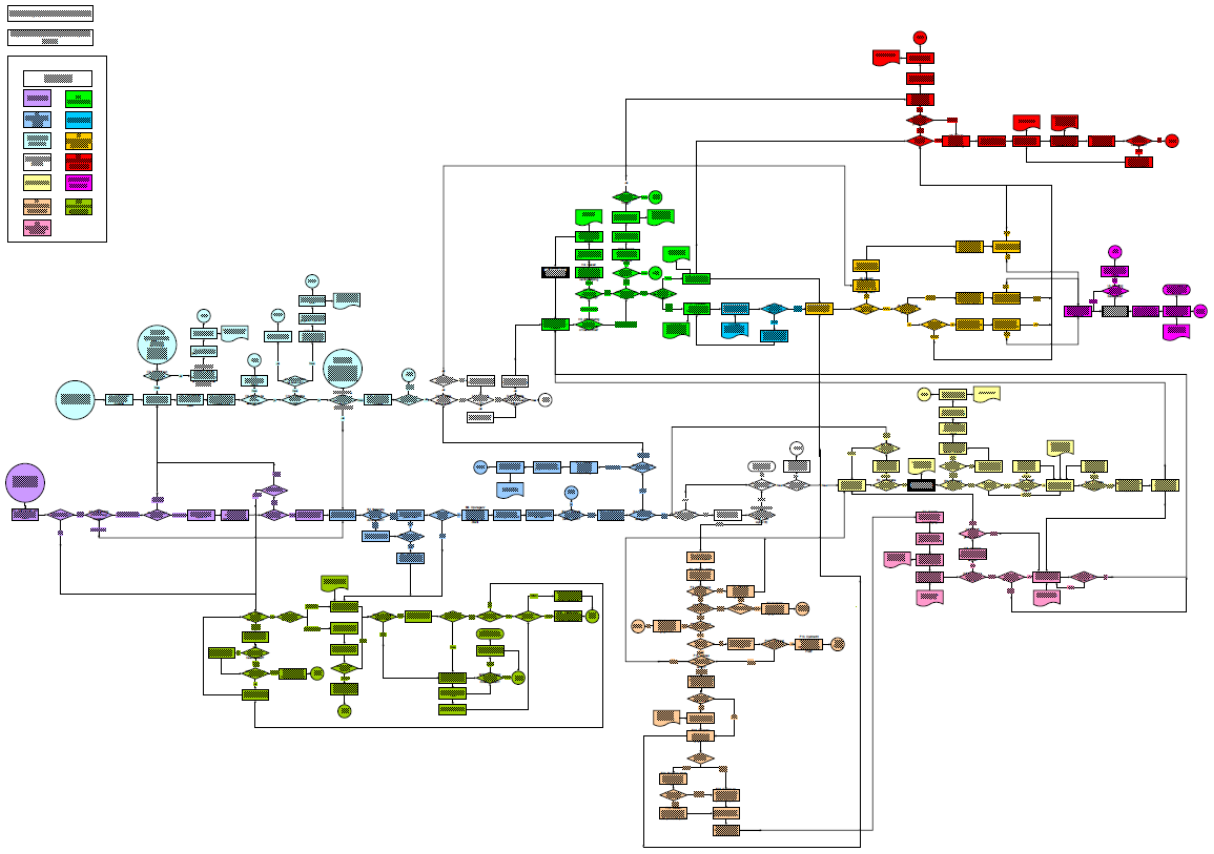


Figure 20 - First version of the large poster

**Issues**

An analysis into the quality of the diagram revealed a variety of issues.

Size

With well over 100 activities and approximately 60 decisions, this diagram is quite large and, given our cognitive limitations, is hard for anyone to comprehend at once. In order to understand what is going on in the diagram, one would need to start at the detail level and work their way through the diagram. The only alleviation is the fact that there are the global process steps which provide a manner of composition thus allowing one to view the diagram from a higher level of detail.

Besides these cognitive implications, the size of the diagram also poses some practical problems. Extensive zooming is required to view the diagram on a computer screen and when printed, a *wallpaper* print size of A2 or larger is required to keep the text readable.

Multiple starting points

This diagram has four starting points while a proper flowchart should have only a single point of entry. Having multiple starting points makes the diagram less deterministic and difficult to navigate, which is already hard enough considering its size.

When we increased the depth of our investigation, we learned that the diagram actually contains multiple processes which are interwoven into the diagram. Besides the primary process flow which is directly related to the primary goals of the project, there also other process flows which span across the various global process steps but serve a different purpose. We argue that a diagram should contain only that information which is relevant for the purpose of the diagram and nothing else.

Structure

Elements are spread seemingly at random and lines cross each other unnecessarily often, which make the diagram difficult to interpret. Keeping a diagram neat and tidy makes it easier on the eyes

as well as the mind. Furthermore, a clean diagram promotes a sense of professionalism which, considering the importance of this specific diagram, should not be neglected. We argue that a simple reordering and realigning of the various concepts would improve the readability of the diagram and make the diagram look more organized. This is also a clear-cut case of the importance of tooling. Professional modeling tools come with advanced algorithms that can reorder the contents of a diagram into a more consistent structure.

Anti-patterns

The diagram contains various recurring patterns of activities and decisions which seem to describe aspects of a process which are difficult to express using the limited syntax of flowcharts or are otherwise questionable. Consider the two partial diagrams as presented in figure 21.

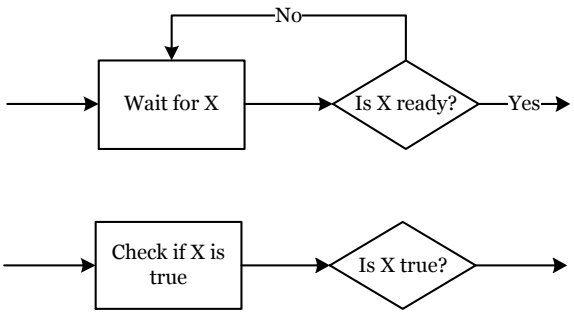



Figure 21 - Two examples of flowchart anti-patterns

The top pattern seems similar to the so-called busy-waiting technique used in software engineering. An activity is used to describe the act of waiting for some information to be received (e.g. receiving a letter). This activity is then followed up with a decision that checks if the information has been received. If so, the process continues on as normal and if not, the process returns back to the waiting state. Clearly, we understand what is being described by the pattern but we argue that this does not belong in a flowchart and this pattern is a consequence of coping with the limitations of flowcharts. More advanced business process modeling languages have constructs that can be used to describe such a pattern. For example, BPMN uses Intermediate Message Events to halt a process until a specific message is received.

The bottom pattern contains a redundant element. An activity is used to check if a certain condition holds and is followed up with a decision that branches on the result of this decision. We understand that the checking of some condition can be described as an activity and might even cost considerable effort, but we argue that this check is implicit to the decision construct. The redundant activity should either be removed or renamed to describe the actual task that produces the desired information required to make the decision. For instance, the decision 'Day or night?' could be preceded by the activity 'Check whether it is day or night', but this would not add much information and would leave it up to the person executing the process how this task should be achieved. A possible solution is to rename the activity to 'Check the time' or 'Look outside'.

Domain knowledge

Many labels that describe the activities and decisions in the diagram contain abbreviations or words from a specific domain vocabulary. This isn't a problem per se, as long as all target users of the diagram are familiar with this domain. Besides the used language, we argue that the diagram is difficult to understand without the presence of a domain expert, preferably one involved with the actual creation of the diagram, to explain what the diagram describes. A possible solution would be to use text annotation to add explanatory information to the diagram.

### 4.2.2 Large Poster – Second Version

The architect entered the project sometime after the creation of the first version of the large poster. He argued that the old diagram lacked overview and that the introduction of varying levels of abstraction would make the diagram more suitable for all involved parties. Thus, a second version was created which is shown in figure 22.

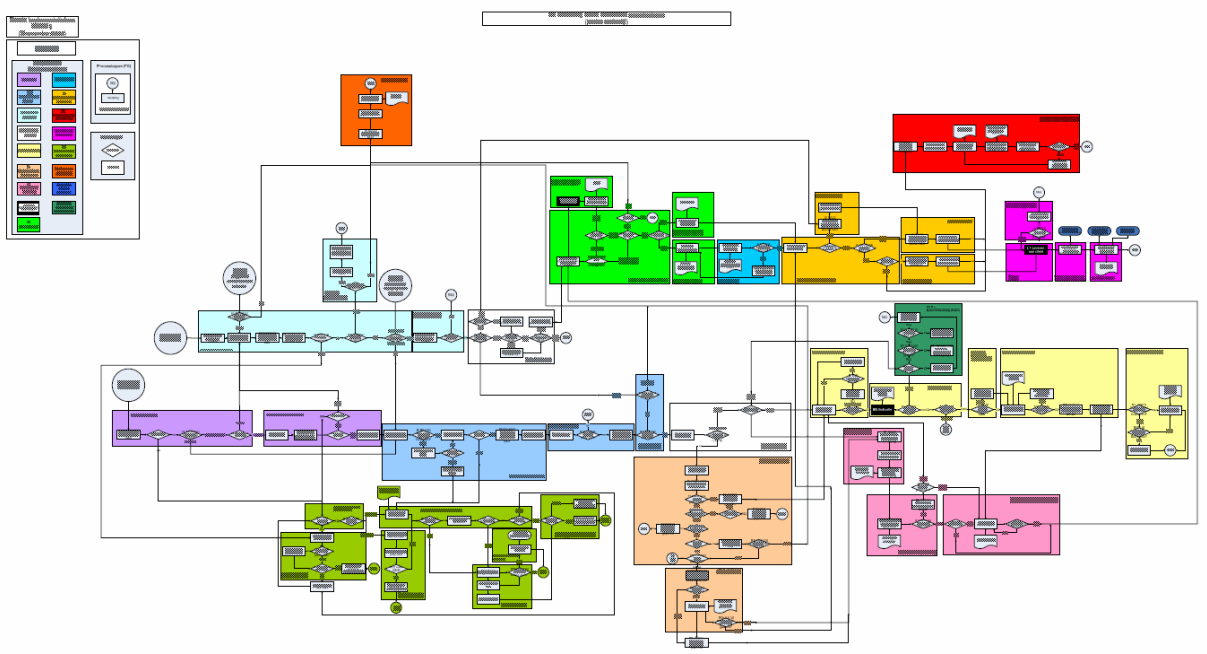


Figure 22 - Second version of the large poster

Various conceptual changes have occurred with the creation of this diagram:

Firstly, the global process steps of the previous diagram are now called *work processes*, which are an ordered set of process steps that are executed within a single organizational unit with the purpose of making a specific contribution to the provision of a service towards a citizen, company or organization.

Secondly, the construct called *process step* has been introduced. A process step is an ordered set of activities which can be executed uninterruptedly by one person or machine within a single business function. These process steps constitute a new layer of abstraction between the work process layer and the activity layer. In comparison, the second version contains three layers of abstraction while the original version only contained two layers.

These conceptual changes have also led to considerable graphical changes. Rectangles are used to group activities and decisions which belong to a specific process step, whereas the color of the rectangle indicates what work process it belongs to. While the purpose of color has remained the same, it is applied in a different manner. To summarize, the used constructs and their abstraction layers are:

- The flowchart shapes represent the activities and decisions which constitute the first layer of abstraction.
- The rectangles that group the flowchart shapes represent process steps which constitute the second the layer of abstraction.
- The color of the rectangles represents the work processes which constitute the third layer of abstraction.

### Issues

The second version of the large poster has some significant changes over its first version, but most of these changes are additions to the diagram rather than modifications of existing elements. In other words, the second version can be considered an extension of the first version and as a consequence, the issues of the first issue are also present in the second version. In addition, two new issues have arisen on which we will now elaborate:

Comprehensibility

Although the introduction of process steps is a clever way of extending the diagram with a new layer of abstraction, we think that it makes the diagram less comprehensible. The more information a diagram represents the more daunting it will be to its viewer; up to the point where he or she is cognitively overloaded and the additional information obstructs the thought process. We do not believe that this is the case, but do argue that a degree of clarity has been lost in when compared to the previous version of the large poster.

Consistency

The addition of new constructs and other alterations have led to various inconsistencies within the diagram. Some constructs are not used in a consistent manner, such as the process terminators which are most often drawn outside of the process step rectangle in a neutral color but sometimes appear inside the rectangle or have adopted the color of the rectangle. Another example is the legend which has been extended with descriptions of the various constructs used in the diagram. The extension of this legend is a valuable addition, as it helps viewer to better understand the diagram. However, the legend is incomplete because the document construct is missing, which is a less popular flowchart construct.

One of the main aspects of modeling is to lay down a set of conventions before starting and to adhere to these conventions during the creation of the model. Such conventions can be anything from strict rules as set by a standard (e.g. a decision must have at least two outbound arrows), simple guidelines (e.g. try to avoid crossing connector lines as much as possible) or anything the modeler introduces on his own (e.g. all activities and decisions must belong to a process step). The more a modeler *plays by the rules*, the more consistent, comprehensible and clear his model will be.

### 4.2.3 Large Poster – Third Version

The third version of the large poster shown in figure 23 was created to serve as yet another abstraction of the business process. The instigator of its creation was the architect whom argued that the second version contained too much detailed information and lacked a proper top-down view of the business process. The third version leaves out the lowest layer of abstraction that was present in its predecessors and focuses on the relations between the work processes present at the highest level of abstraction. Note that the third version accompanies the second version, but does not replace it. Both versions serve a different purpose and consequently fulfill different demands for information. This version was reluctantly accepted by the stakeholders of the municipality.

An investigation was performed on this diagram to find out what changes were made and how these changes affected the various stakeholders.

First and foremost, all elements at the lowest level of detail such as activities, decisions and terminators have been collapsed into their corresponding process steps. In essence, the first layer of abstraction has been removed, which means that this diagram no longer contains any flowchart constructs and thus can no longer be considered a flowchart. By removing this layer, the total amount of elements represented in the diagram has been reduced to a quarter of those used in its predecessors, which makes the diagram a lot easier to grasp. From a theoretical perspective, the removal of the flowchart layer is a clear-cut case of abstraction through information hiding as discussed in section 2.6 and subsection 2.7.6.

Secondly, a new construct, the trigger, has been introduced at the work process level. A trigger is a construct used to describe the information flowing from one work process to another. Up until now, relationships between two elements were depicted by an arrow, which indicated some kind of causal relation, but revealed no further details on the nature of the relationship. As each work process concerns a separate organizational department with different employees, some kind of information exchange needs to take place when a process instance flows from one work process into another.

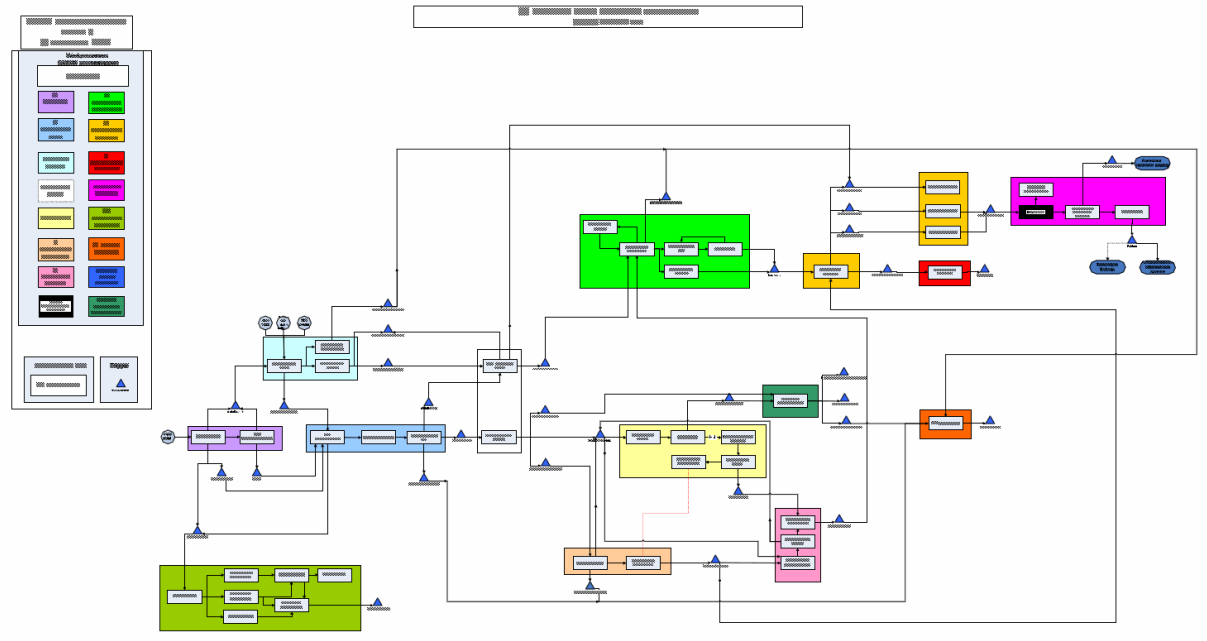


Figure 23 - Third version of the large poster

The term 'trigger' is used to indicate that a quantifiable piece of information (e.g. a document, e-mail or letter) is responsible for invoking a work process. In the business process modeling languages we covered in subchapter 2.1, such a construct is most often referred to as an 'event'.

**Issues**

An analysis of the third poster was conducted which revealed the following issue:

Non-deterministic triggers

Most work processes have a specific process step that serves as the point of entry for that work process and this process step usually requires a single predetermined type of information. As a result, relationships between this work process and other work processes have a similar trigger. To save space and reduce redundancy, these relationships are depicted using a single trigger in the diagram. Although this means that fewer elements are used to achieve the same description, it also reduces the clarity of the diagram. In one particular case shown in figure 24, a trigger has multiple inbound *and* outbound relationships, which implies that any combination of inbound and outbound flows is valid. However, we found that this was not the case.

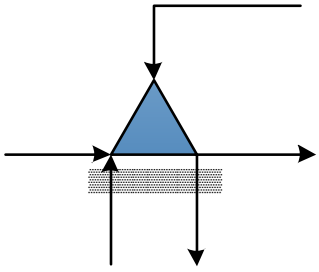


Figure 24 - Trigger with multiple inbound and outbound flows

4.2.4 BPMN Model

When the requirements analysts became involved in a follow-up project, they used the various existing process designs as a basis for a new model. In contrast to all previous modeling efforts, the requirements analysts used a popular business process modeling language (BPMN) in combination with a professional business process modeling tool. The end result was a model consisting of a collection of diagrams at various levels of detail. Figure 25 shows the diagram at the highest level of detail, from which one can navigate to the other diagrams at increasingly lower levels of detail.

As we mentioned in the outline of the case, the model was rejected by some of the stakeholders of the municipality because they longer recognized their business process. Besides the different modeling language and style, the most notable change is the use of decomposition, resulting in a set of interrelated diagrams rather than one single diagram.

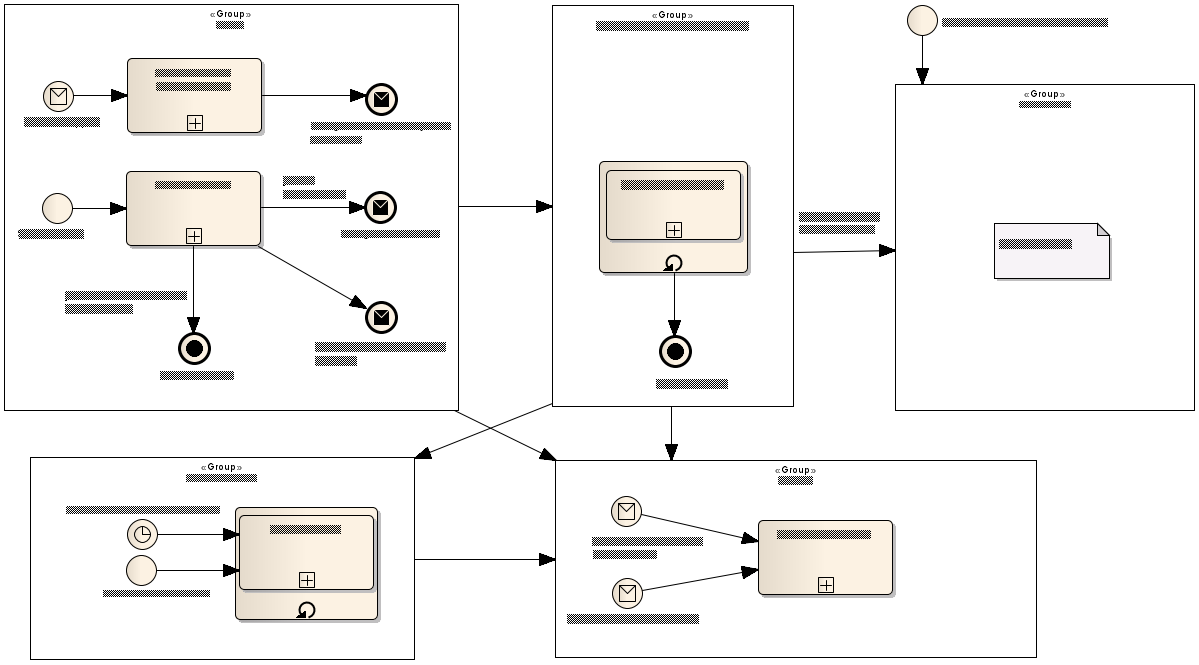



Figure 25 - Root diagram of the BMPN model

The set of diagrams is built like a tree, with a main diagram at the root of the tree to provide an overview of the overall business process. In the root diagram, the distinction between the various main business functions is made through the use of so-called group-boxes. Each group-box contains one or two sub-processes that represent the business process which fulfills this business function. These sub-processes can be expanded, by means of decomposition, to reveal their details.

Figure 26 shows a tree-view of the set of diagrams and was created to see how the various diagrams related to each other. From this view, it is apparent that the height of the tree is four. There are five diagrams present at the first depth level of which only one diagram has any descendants. This implies that this specific process is more complex than the other four processes which makes sense considering this particular process is considered the *main process* of the five processes.

**Issues**

An analysis of the various diagrams was conducted which revealed the following issues:

Activity-Decision inconsistencies

The most profound issue discovered were the incorrect use of various basic BPMN concepts. In several diagrams, there are one or more activities which actually describe a decision. As a result, these activities have two or more outbound control flows and it is not clear which control flow should be followed under what condition.

In another case, a single decision is used to describe two or more decisions. While this makes the diagram more compact, it also reduces the clarity of the diagram, because the viewer has to spend additional effort in order to unravel the compound decision in his mind. Figure 27 shows two examples of these inconsistencies.

This mix-up of concepts which are fundamental to BPMN, or flowcharts in general, is troublesome. However, we must add that although these modeling patterns are incorrect, they are still quite understandable for the viewer. Moreover, as we will show in 4.3.3, the BPMN language is partially to be blame for these inconsistencies.

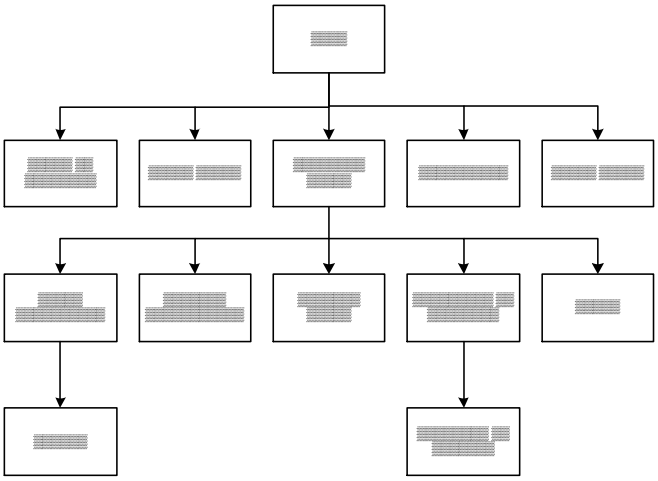


Figure 26 - Hierarchical view of the various BPMN diagrams

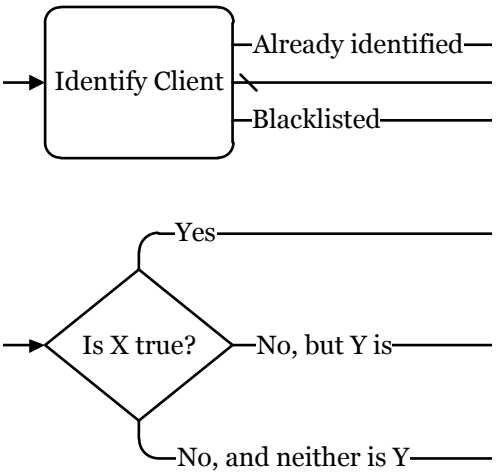



Figure 27 - Incorrect use of basic BPMN constructs

Simulated process orchestration

The sub-processes in the root diagram are not connected in a normal BPMN fashion. Instead, a construction with group-boxes is used to link one or more sub-processes to a core business function. Additionally, arrows are used to represent a causal relationship between these core business functions. This construction has pragmatic value as it gives a rudimentary view of the overall process flow to its viewers. However, it lacks semantic value as group boxes in BPMN are an annotation construct. From a strictly semantic point of view, the root diagram represents a collection of unconnected sub-processes. Although the diagram appears to show a causal relationship between the various sub-processes, it is unclear what events from one sub-process invoke the other.

We argue that this root diagram provides no real added value as a BPMN diagram and should either be fixed by connecting the various sub-processes in a normal BPMN fashion or be replaced by a different type of representation such as a process landscape.

Lack of common BPMN constructs

While the model certainly has more expressive power than the large poster, some common BPMN concepts are only rarely used. Consider the issue of the Activity-Decision inconsistencies. In BPMN diagrams, multiple outbound arrows would usually originate from an Exclusive or Parallel Gateway, rather than an activity. However, this common construct is only sparsely used in the BPMN diagrams we analyzed. The lack of common BPMN constructs seems somewhat contradictory with the intentions of the requirement analysts. They explicitly choose BPMN over flowcharts, since flowcharts lacked the expressive power they needed to model the business process. Yet, in their application of BPMN, they use only a limited part of the BPMN vocabulary.

Besides these issues, there are some advantages this model has over the large poster. For instance, most connectors have been given a description which is only rarely the case in the large poster. Although this makes the model a little bulky text-wise, we found that the addition of a description in natural language to most elements increases the clarity of the model, thus making it easier to understand. The use of some BPMN constructs such as Message Events also more accurately indicate what the input or output is of a certain process. Unlike the flowchart based posters, the BPMN model is not limited to describing processes as a sequence of activities and decisions, but also gives a rudimentary representation of the input and output of these processes.

## 4.3 *Conclusions*

Based on the analysis of the models of this case, various conclusions can be drawn regarding the use of models, the role of different modeling languages and how these relate to the purpose of different parties involved with the modeling activity.

### 4.3.1 Definition of overview

There is a conflict of opinion between the stakeholders of the municipality and the requirements analysts on what constitutes as an *overview*. The prior party believes that the original large poster provides the best overview while the latter party argues that their BPMN model provides the best overview. Two distinct definitions of overview can be inferred from this observation:

- From the perspective of the stakeholders of the municipality, overview is achieved by creating a single representation with a maximal level of detail. They argue that all information should be represented in this single model because in this manner no information is left out, thus guaranteeing a maximal hold on the process. To abridge, the fact that all information is represented in a single location creates the overview.
- From the perspective of the IT stakeholders, overview is achieved by creating a greatly simplified representation with a minimum level of detail. They argue that the human mind is incapable of grasping all the little intricacies of the complete business process at once, thus we need to create abstractions in order to cope with this complexity. The concepts used in these abstract representations are elaborated using other more detailed representations thus allowing one to zoom in on a particular part of the business process when needed. To abridge, the fact that there is a main model at the highest level of abstraction that contains little or no details creates the overview.

This fundamental difference also carries implications for the way these two parties create models:

- When the stakeholders of the municipality create a model, they start at a maximal level of detail and continue to work at this level of detail until they are satisfied that the model contains all the relevant information. We call this a *bottom-up* approach towards modeling.
- When the IT stakeholders create a model, they start at a minimum level of detail and create additional models at an increasing level of detail until they are satisfied that the overall collection of models contains all the relevant information. We call this a *top-down* approach towards modeling.

We must add that in this particular case, the approach of the stakeholders of the municipality can be considered *bottom-only* rather than bottom-up, because their focus mostly remained on the details of the business process. It was the architect who emphasized the use of layers of abstractions in the large poster.

Neither approach towards process modeling can be considered better than the other, nor can we conclude that a series of interrelated models is better than one large wallpaper-sized diagram. However, if the stakeholders of the large municipality prefer one large diagram containing all the details, then it should be possible to aggregate the set of diagrams created by the requirements analysts into such a diagram. Some professional modeling tools allow the creation of such a representation, which is called a *wallpaper view*, *exploded process view* or *flat process view*. The use of such a feature would allow both parties to create a useful representation of the business process based on a single source of information. In essence, these representations would be different views based on one model. Unfortunately, such a feature was not supported by the tools used in this case.

### 4.3.2 Suitability of flowcharts

Some of the issues present in the large poster can be traced back to the used modeling language: *flowcharts*. This language was explicitly chosen by the stakeholders of the municipality due to its ease of use and understandability by people whom are unfamiliar with business process modeling. We agree that from this point of view, a *lightweight* modeling technique is the appropriate choice and flowcharts fit this choice well. These types of models are easy to draw, understandable for novices and are straightforward to communicate with.

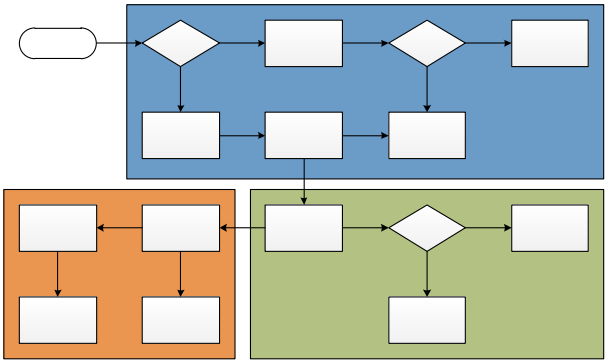


Figure 28 - Combination of flowchart notation and colored rectangles

The broad applicability of flowcharts makes them appropriate for the modeling of any process that involves steps and choices. However, as we mentioned in 2.2.1, the technique itself only describes a, very limited, set of constructs without any clear standardized rules on how to use these. We argue that this lack of rules is what makes flowcharts such a widely applicable modeling language as it leaves plenty of room for its users to be creative and adapt their own style during the modeling process. In the case's flowchart of the large poster, collections of activities and decisions are grouped to form so-called work processes, which are represented by colored rectangles. Neither the concept of a work process nor the use of colored rectangles – or color for that matter – is inherent to the flowcharting technique. These constructs were an invention by the business analysts who created the model as they, apparently, found this an adequate way to represent this information.

From the viewpoint of ontological expressivity as discussed in subsection 2.5.1, flowcharts suffer from construct deficit, due to the small set of available constructs. This deficiency is somewhat alleviated by the business architects who created their own constructs. Moreover, the lack of semantics makes it difficult to determine what the existing flowchart constructs describe, which seems indicative of construct overload. The lack of semantics grants the user considerable freedom of expression, but also constitutes the fallacy of flowcharts: They are *too* flexible.

Performing a search query on the word *flowchart* on the internet gave us access to an abundance of flowcharts. The great diversity in the shapes and sizes of these flowcharts strengthens our conviction that, while flowcharts are very useful for the description of a wide range of real world processes, they lack a solid framework for their creation. We extended our query to an ad-hoc analysis in order to see which of these flowcharts were most comprehensible. At a glance, flowcharts consisting of no more than 30 concepts with a single starting point and describing nothing but steps and decisions provided the best clarity and were most easy to understand.

The style of the flowchart was also found to be of importance. For instance, some flowcharts gave each construct a unique color. Initially, this seems to be redundant as each construct already has a distinct shape to differentiate itself from other constructs. However, the addition of color made it easier to differentiate between various constructs and also made the flowchart more pleasant to look at.

When compared with the issues present in the case's flowchart, we observe that most of the flowcharts found on the internet suffer from one or more of the encountered problems. Specifically, issues regarding the size and structure were present in a multitude of the flowcharts. Without proper tooling and use of decomposition, flowcharts can easily grow in unmaintainable and incomprehensible diagrams. To conclude, modeling should enable one to cope with the complexity of a system, not struggle with it.

### 4.3.3 Suitability of BPMN

An in-depth analysis revealed that the issues found in the BPMN model can be partially traced back to deficiencies in the BPMN standard.

Firstly, it was shown in subsection 4.2.4 that the root diagram was not a conventional BPMN diagram and that there was no sense of a single end-to-end business process. Top-level processes tend to be very generic business processes which run over a long period of time and involve multiple stakeholders and information systems. In contrast, normal BPMN activities are usually performed by one stakeholder within a small period of time (e.g. OTOPOP principle from subsection 3.4.3). Because BPMN aims to provide representations of business processes at the detail-level, it is less suited for describing business processes at the high-level. Although sub-processes allow one to decompose a business process, these do not introduce any conceptual changes.

Secondly, the conceptual overlap between normal BPMN tasks and sub-processes can lead to ambiguities. For instance, when we inquired why a certain task in a BPMN diagram had multiple outgoing control flows, we learned that the task was actually a 'to be specified' sub-process. Tasks are allowed to have multiple outgoing control flows, although it is strongly recommended that these control flows are labeled to specify under what conditions they are activated. Without these labels, it may not be clear how a process instance proceeds after the task has been completed.

As shown in subsection 3.4.4, decomposition of a task into a sub-process adds additional constraints to the sub-process, which can be best satisfied by upholding the Single Point of Entry principle. Multiple inbound control flows are not allowed and multiple outbound control flows are strongly discouraged for aforementioned reasons.

### 4.3.4 Purpose of modeling

Different stakeholders have different goals when they are modeling [52]. The three versions of the large poster from the case are a good example of a clash between different parties with different goals. The architect wants to design an information system while the stakeholders of the municipality want to redesign a way of doing work. Neither goal will lead to a model which is wrong nor would it be wise to pick one goal over the other. Each purpose demands different information from the diagram, which subsequently means that either a single diagram should contain *all* this information or many diagrams need to be created, each for a specific purpose. In our case, both approaches have been applied as the first version of the large poster led to the creation of two new versions. Unfortunately, none of these efforts resulted in a product that satisfied the goals of all stakeholders.

Both approaches have their merits and their pitfalls. It is easier to create and maintain one diagram, but each stakeholder will have to place more effort in understanding the diagram. Additionally, it might be challenging to choose a notation which is suitable for all stakeholders (e.g. flowcharts or BPMN). On the other hand, creating multiple diagrams will more effectively satisfy the information demand of all stakeholders, but requires more effort to create all these diagrams. Moreover, having multiple diagrams introduces the job of keeping all diagrams in sync. A change in one diagram must be properly translated to a change in all other diagrams. Adequate modeling tools can assist in maintaining consistency across diagrams. The idea is to create multiple diagrams based on a single source of information (a.k.a. views on models principle). In contrast, lack of synchronization across diagrams effectively means that there are multiple models. This can lead to a state where different stakeholders uphold different versions of the business process, which impedes the creation of mutual understanding.

A complicating factor for business process modeling is that most of the information lies with the business stakeholders; in our case, the stakeholders of the municipality. Other parties, such as the requirement analysts and the architect, would like to see these stakeholders create the right model for them. Of course, it is often hard to provide a convincing argument to these stakeholders why they should commit considerable time and effort in something that seemingly does not directly benefit them.

### 4.3.5 Types of abstraction

The three versions of the large poster were created through the use of *conceptual abstraction*, which we discussed in subsection 3.3.2. The first version of the large poster has two conceptual abstraction layers, namely a flowchart layer, describing activities and decisions, and a work process layer, describing processes with significant outcomes. The second version separates these two layers with the addition of the process step layer, which describes a series of activities that can be executed interruptedly by one person. The third version reverts back to two conceptual abstraction layers by hiding the flowchart layer.

The BPMN model was created through the use of *scoping abstraction*, which we discussed in subsection 3.3.2. By using sub-processes, BPMN diagrams are collapsed into activities which in turn can be used in other BPMN diagrams. This might seem similar to the third version of the large poster, in which all constructs at the flowchart layer have been collapsed into their respective process steps. However, in the large poster, each layer of abstraction denotes changes at the conceptual layer as constructs on different layers of abstraction have different semantics. In contrast, in the BPMN model, each layer of abstraction still uses the same constructs, which is indicative of scoping abstraction. In other words, the various versions of the large posters describe different types of information, while the various diagrams in the BPMN model describe the same type of information. As we discussed in subsection 4.3.3, the lack of conceptual difference between BPMN diagrams impedes the design of an appropriate end-to-end process in the root diagram.

To conclude, scoping abstraction is useful for decomposing a large business process, while conceptual abstraction is useful for creating a meaningful process hierarchy. The latter form of abstraction will lead to a more meaningful decomposition, but also demands more effort from the modeler to achieve. In contrast, scoping abstraction is straightforward to apply and could even be performed by a computer.

# 5 General issues

A series of generic issues that inhibit the effective use of business process models can be extracted from the findings of our research and case analysis. Other research which has aimed to achieve similar goals has identified one or more of the issues we found, strengthening the confidence in our findings [3]. For each issue, we attempt to explain its cause with the theories discussed in chapter 2 and chapter 3.

## 5.1 *Conflicting mindsets of involved stakeholders*

We already mentioned that business and IT stakeholders differ greatly, even going so far as to claim that they speak different languages or live in separate worlds. One of the fundamental causes of this separation is the two distinct mindsets of these two groups of stakeholders, which are the result of grossly different educational backgrounds and interests [19]. These differing mindsets also have their implications in the manner of which these two parties create, interpret and experience models.

For example, consider the use of written text versus the use of diagrams to represent information. Given the choice, business stakeholders most often prefer written text over diagrams as written text, in their view, is easier to use and leads to more concrete descriptions. In contrast, diagrams are often perceived as being too abstract. The more formal a graphical representation of a model is, the more it will look and feel like a mathematical equation and the more likely the business stakeholder will lose interest. Another, more extreme, example are jurists who tend to over appreciate written text and their perception of text, be it a legal text or the description on a milk carton, is very different from other people. They will hold a much greater value on text and be much more critical of its contents.

The exact opposite is the case with IT stakeholders as they tend to have an aversion for written text and appreciate graphical representations. Software documentation, especially those documents concerning the technical architecture of an application, contain numerous diagrams such as class diagrams, sequence diagrams, activity diagrams, state diagrams, use case diagrams and so on. To generalize, if an IT stakeholder can replace a piece of text with a diagram, he will often be inclined to do so, as he considers written text to be ambiguous, redundant and inefficient [53].

The use of written text versus the use of diagrams can be mapped to our discussion of natural language versus artificial language in section 3.1. Natural language is the language most people are comfortable with, whereas artificial languages are only used by a minority of people. IT stakeholders are usually well experienced with the use of artificial languages while business stakeholders are not. This issue can be mitigated by adding descriptions in natural language to diagrams. From the perspective of formal modeling languages, text annotation constructs carry little semantic value. Yet, they can have considerable pragmatic value in terms of making the diagram more understandable for business stakeholders.

To conclude, the differences in mindsets between business and IT stakeholders can inhibit the effective use of models. Business process diagrams should balance the use of natural text and graphical modeling constructs in order to create a model that is understandable to both stakeholders.

## 5.2 *Conflicting goals of business process modeling*

As mentioned in the introduction, business process models can be created to describe, analyze or enact business processes. However, these three goals pose a conflict by themselves. Creating a shared understanding of business processes emphasizes the need for a clear and intuitive modeling language. All types of stakeholders should be able use such a modeling language to create or interpret business process models. In contrast, creating computer executable business processes

emphasizes the need for a formal modeling language with additional formal semantics of its execution.

As discussed in section 3.1, it is easier for most people to describe something in natural language than in formal language. Yet, as shown in section 3.2, computers are unable to cope with the ambiguities of natural language. Thus, creating understandable models impedes the creation of executable models and vice versa.

Executable models must adhere to the formal syntax and semantics of the used modeling language. This level of formality considerably constrains the expressiveness of the modeling language and can lead to changes in the model that seem quaint to business stakeholders. Moreover, these rigorous constraints can be experienced as a nuisance by stakeholders who do not understand their importance. The modeling activity is reduced to a game of adapting models to fit the alleged limitations of the modeling language. While these constraints can lead to considerable changes in the model, these changes are not actually translated to the real world business processes. In essence, the end result is a model which poorly reflects reality and means little for the creation of shared understanding between business and IT stakeholders.

To conclude, the three goals of business process modeling are partially incompatible and attempting to fulfill two or all three goals may lead to a model that does neither well.

### 5.3 *Fallacy of imperative reductionism*

Although there are many different dimensions to a business process, most business process methodologies assume an imperative approach towards the description of business processes. Indeed, the aim of business process modeling languages such as BPMN, EPC and RAD is to allow stakeholders to create a model of the business process in terms of a series of tasks. In other words, the shared ideology of all these languages is that a business process can be reduced to a sequence of activities. Such an approach works well for processes consisting of distinct activities in a specific logical order (e.g. the behavior of an ATM machine or the process chain of a car factory).

Unfortunately, real world business processes can be volatile, unpredictable and perhaps most importantly: unordered [54]. Consider a soccer match as a business process. The process involves a predefined set of stakeholders, that each has one or more assigned functions. There is shared understanding between the stakeholders on what goal should be achieved and all stakeholders will contribute in some way to the achievement of this goal. A set of constraints is in place that limits the stakeholders in how they can achieve their goals and finally there are metrics in place to check if the goals have been reached.

Although all this information is known, it would still be difficult to capture the actual process of the soccer match in a flowchart or BPMN model. The imperative nature of these methodologies forces the modeler to not only specify what the process is, but also how it is executed. As we discussed in subsection 3.1.3, problems can occur when the purpose of the modeling language and its actual use are misaligned. All the information provided by the soccer match example relates to what the process is, but reveals no details on how it should be performed. This makes sense as each soccer match is unique and the activities performed in its process are chaotic and unordered. Attempting to create a representation of this process as a sequence of activities and decisions will lead to an over specification of the process, resulting in a rigid design that poorly matches reality [55], [56].

In contrast, declarative languages attempt to tackle this issue by allowing programmers to specify *what* the program should accomplish without having to specify *how* it should do this. Unfortunately, the development of declarative business process models is still in its early stages [57]. Alternately, one can use a modeling language that does not specify temporal sequences (i.e. IDEF0).

To conclude, unordered business processes are not suitable to be represented in an imperative process language and any modeling effort that aims to capture such a process in an imperative modeling language will result in a model that inadequately reflects reality.

### 5.4 *Novice modelers versus expert modelers*

Business process modeling places considerable demands on its users. Ideally, its users should have knowledge of business process management, business process modeling methodologies and supportive modeling tools. Moreover, good abstraction and conceptualization skills will help these users to translate the real world into adequate conceptual representations.

In practice, these demands are rarely met by the people involved in the business process modeling activity. This is not at all surprising as BPM projects involve stakeholders from a variety of disciplines, most of which are not concerned with conceptual modeling. As a consequence, two types of modelers are involved with the design of the process model [58]:

- *Novice modelers*, whom are inexperienced users of process modeling methodologies and tooling. Business stakeholders can generally be considered to belong to this group.
- *Expert modelers*, whom are experienced users of process modeling methodologies and tooling. Business process analysts and IT stakeholders can generally be considered to belong to this group. IT stakeholders represent a unique group of modelers as their IT-background incorporates modeling.

A process modeler will adapt a different approach depending on whether he or she is a novice modeler or an expert modeler:

- Novice modelers tend to prefer lightweight modeling methodologies such as flowcharts, while expert modelers tend to prefer mature business process modeling methodologies such as BPMN or EPC. When a novice modeler uses BPMN, he or she will have a tendency to only use the most common BPMN constructs. In contrast, expert modelers use a much larger part of the set of BPMN constructs [18].
- Novice modelers tend to prefer user-friendly modeling tools, which allow them to create a model right away. These tools include the use of whiteboards and pen and paper. Expert modelers tend to prefer elaborate modeling software.
- Expert modelers are comfortable with handling large and complex models, while novice modelers are not so.

To conclude, stakeholders can be divided in two groups, namely novice modelers and expert modelers. Each group takes a different approach towards the creation and interpretation of business process models.

### 5.5 *Quality of abstraction*

Bad abstractions in business process modeling can be particularly dangerous as they are not straightforward to detect, promote an inefficient approach towards understanding business processes and can ultimately lead to inadequate process designs. Any problem solving strategy involves a degree of abstraction with more complex problems generally requiring higher levels of abstraction to be solved. In the corporate world, where both cost and time are a valuable commodity, employees are more inclined to pick the less efficient but straightforward solution over the proper solution which takes more time to think out.

Consider the application of conceptual abstraction as discussed in subsection 3.3.2. When applied, a detailed business process can be generalized into several layers of abstraction, each new layer more generic than the last. The layers used in the case (i.e. work processes, process steps and actions) are a common way of abstracting from detailed work instructions to generic goals of the organization, which is a sensible approach towards understanding a business process. An example of a less sensible approach is to generalize activities based on their location. While this may be useful in creating an overview of what happens where, using it as a criterion for conceptual abstraction can lead to a rigid design of business processes.

Additionally, having many levels of abstraction does not guarantee a good solution. Each level of abstraction forms a barrier one has to cross using their intellect and ability to give meaning to abstract concepts. Having many levels of abstraction may pose advantageous to those able to grasp all these levels, but can also severely inhibit comprehensibility of the model when others are unable to reach this level of abstraction.

For example, software developers are keen problem solvers and often enjoy creating good abstractions. Unfortunately, these thinkers may also get lost in their struggle to find the most beautiful and optimal abstraction. Efforts that attempt to create an abstraction that encompasses everything usually end up with designs that are too broad to be applicable.

To conclude, models that suffer from bad abstraction tend to be correct models as they obey all the rules as laid out by the modeling language, but they capture the actual business process in a non-straightforward manner.

### 5.6 *Scientific approach versus business approach*

As we mentioned in the introduction, BPM has received much attention from both the scientific community and the corporate world. It should come as no surprise that both fields take a radically different approach towards the subject.

Business process modeling languages created by the scientific community tend to be based on mathematical notation and formal language. An example of such a language is Petri Net. These languages enable one to describe business processes in a formalized manner, which can subsequently be mathematically analyzed or interpreted by a computer. As we discussed in 3.1, formal languages are less straightforward in use than natural language. Additionally, business stakeholders generally lack the appropriate expertise required to use formal languages. As a result, these languages are experienced as being too difficult to use and are thus deemed unsuitable for use in practice [59].

Business process modeling languages created by the corporate world tend to be based on practical experience with business processes and the necessity for such a language to arise. Examples of such languages are RAD and IDEF. These languages are created with their practical application in mind, but tend to lack the formality of the ones created by the academic world. As such, these languages are suitable for creating models to achieve shared understanding among all stakeholders, but less suitable for mathematical analysis or computer interpretation.

Although it would be cliché to conclude that scientific endeavors result in solutions that lack practical value, we do believe that the scientific community should place more emphasis on topics such as effective knowledge representation and conceptual modeling when constructing business process modeling languages.

### 5.7 *Lack of standard business process modeling language*

As shown in section 2.2, there are various business process modeling languages and they all take a unique approach towards the creation of business process models. Some languages emphasize expressiveness by presenting a broad range of constructs, but lack clear guidelines on how to apply these constructs. Other languages have a limited set of constructs, but incorporate a level of formality, thus limiting ambiguity and enabling computer interpretation. Some languages emphasize the creation of business understandable models, while others emphasize the creation of computer executable models.

None of these approaches can be deemed incorrect, nor is it straightforward to deduce that one language is more suitable for business process modeling than another. The fact that there are numerous business process modeling languages indicate to some extent that it is challenging to create a one shoe fits all language. Yet, the lack of a single concise standard makes it difficult for practitioners to pick the right methodology and hinders the development of business process

management on the whole. In practice, it is common for BPM projects to use two or more business process modeling languages [52].

At the time of writing, it would seem that BPMN is the best candidate at becoming the standard for business process modeling. While we support the creation of this standard, we must point out that BPMN takes a specific direction towards business process modeling which may result in models that are not suitable for business stakeholders in a BPM project. We will elaborate this major deficiency in section 5.9.

To conclude, a complete and well documented business process modeling standard will help modelers capture the relevant details of a business process in a concise and systematic manner.

### 5.8 *Lack of adequate tool support*

The use of proper tooling is of particular importance with business process modeling as these models tend to contain a vast amount of information and serve a multitude of purposes including description, analysis and enactment. Unfortunately, current professional process modeling tools tend to be elaborate diagram drawing tools rather than actual model creation tools.

For example, a common problem we encountered with the use of process modeling tools is the support for viewing a model at different levels of abstraction. Business process models tend to grow rapidly, both in size and in diversity of information, yet most process modeling tools only have primitive ways for coping with these large models. In many BPM projects, including the ones of our case, the center of attention is a wallpaper size diagram that contains all the information. Such a large diagram may have its uses, but also poses several problems, which we mentioned in 4.2.1. Given that business process modeling is supposed to reduce complexity and increase understanding, modeling tools should support these goals. Large process models are a fact of most BPM projects and tools should enable users to manage them, rather than struggle with them.

Model validation is another example of a feature which current process modeling tools could improve on. Process modeling languages such as BPMN are endowed with semi-formal semantics, which means that they can at least be partially validated by a computer algorithm. Although several process modeling tools we have used support this feature, they do so modestly as they are limited to detecting only the most obvious errors (e.g. unconnected or unlabeled elements).

The features we discussed in the previous paragraphs are present in only the most advanced modeling tools, which are aimed at the IT stakeholders. While these tools can be very powerful, they are also quite unsuitable for business stakeholders as they tend to be overloaded with buttons, toolbars, menus and windows. Considering that business process modeling concerns both business and IT stakeholders, we argue that the supportive tools should be usable by both parties. In other words, the lack of tool support can also be interpreted as the lack of *user-friendly* tool support.

Finally, various process modeling tools interpret standards such as BPMN differently. Arguably, this is a consequence of ambiguity and incompleteness of the BPMN specification. As business process modeling standards such as BPMN become more popular and the amount of supporting tools increases, the importance of a clear and unambiguous standard increases as well. Consider the history of standards-compliance of web browsers such as Internet Explorer, Firefox and Safari, the great browser war and how this impacted the world of web-design. We would not like to see this history repeat itself with business process modeling standards and its supportive tools.

To conclude, modeling tools cannot check whether a model reflects reality, but they can validate a model against the syntactic and semantic rules of the used modeling language.

### 5.9 *Presence of technical details in business process models*

Business stakeholders use common sense notions of behavior and time and do not think computationally. Fundamental principles from the field of computer science, such as parallelism and concurrency, are unknown to these stakeholders and any choices made during the business process design phase that concerns these principles will seem strange to them.

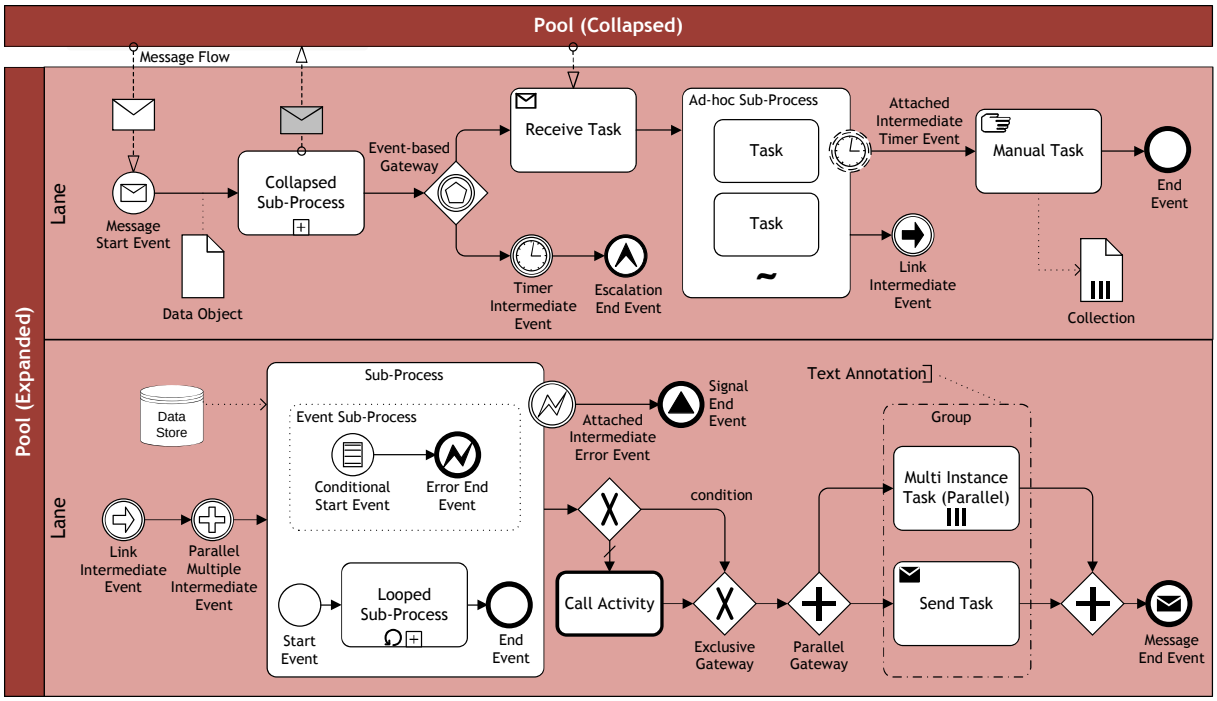



Figure 29 - Specification of BPMN 2.0 Collaboration Diagram. Reprinted from the "Berliner BPM-Offensive"

Although some of these principles may be explained using business friendly analogies, most of these principles remain difficult to understand without a computer science background. If the idea of business process modeling is to empower the business stakeholders, then these methodologies should adhere to the expertise of the business. Yet, most business process modeling languages shown in section 2.2 are littered with technical terminology (e.g. AND/OR/XOR-gates). Although executable process model languages try to abstract from the technical details, they do so poorly and choices made at the technical level often have consequences for models at the business level. This is an unavoidable consequence of the principle of leaky abstraction, which we discussed in 3.3.1.

At the time of writing, BPMN 2.0 has just been released. A quick study of its new features shown in figure 29 reveals that the focus has been placed on adding more detail to the specification in order to make the models more suitable for execution. This is great news for the IT stakeholders who are concerned with the technical support of BPM projects, but means little for the business stakeholders who should be in charge of the actual design of the model. In fact, we believe that this development of BPMN makes the standard less suitable for the business stakeholders. We argue that if BPMN continues on this road, it will become a visual programming language and business ownership of BPMN models will be lost. Business stakeholders might still create initial versions of their processes, but IT will be burdened with the task of fixing these models until they are executable, at which point the business stakeholders will no longer understand the models. To summarize, the whole notion of collaborative process modeling is effectively lost.

We believe that the BPMN standard should at the very least make a distinction between information related to the execution of a model and the information not related to the execution of the model. The prior type of information is of a technical nature and has little to do with the actual business logic contained within the model. The principle of information hiding discussed in 3.4.6 can play a crucial role in achieving this separation. Technical details that make a process model executable are only of interest for IT stakeholders who want to enact the model, but these details will most likely confuse all other stakeholders. The unfortunate truth is that the technical details might have significant implications for the process models and thus for the business stakeholders, which brings us right back to the original problem of business-IT alignment as discussed in 2.3.

To conclude, technical details present in business process models tend to distract its viewers from that which these models are trying to capture; the business process.

### 5.10 *False expectations of BPM projects*

Business process reengineering, software selection, software development, customizing of standard software, workflow specification, simulation, animation, human resource planning, activity-based costing, project management, knowledge management, benchmarking, certification, continuous process management [60]. These are all motivators for engaging in BPM and with this growing list of goals the expectations of BPM grows with it.

Popular strategic goals associated with BPM are continuous process improvement, transformational gains, incremental gains and substantial benefits. These vague yet catchy terms give the impression that by doing BPM, employees will become more productive, costs lowered, risk reduced and control increased. Although these are all possible outcomes of a BPM project, they are by no means straightforward to achieve. Unfortunately, many BPM projects are launched without first determining the goals of the project, creating a clear strategic agenda and determining what processes will be targeted. As a result, these projects often fail to deliver a desirable outcome and BPM is falsely accused of being inadequate.

Business process modeling is also a victim of false expectations. BPMN, for instance, is often portrayed as a solution while in essence it is only a tool to help people map their business processes. Whether an actual advantage is achieved relies on a great variety of factors that have nothing to do with the BPMN standard. As we mentioned in 2.1, the modeling activity is only one part of the complete BPM life cycle. Without proper change management, the created models of 'to-be' business processes will remain elaborate drawings rather than becoming blueprints for real world business processes.

Another false expectation of process modeling is that one model can fulfill all three goals of process modeling (e.g. description, analysis and execution). Executable process models contain a large amount of details regarding the technical implementation, often obscuring that information which is relevant to business. In contrast, business-friendly process models are usually created in a non-formal modeling language and are therefore unsuitable for process simulation or enactment. As we already discussed in section 5.2, attempting to achieve all these goals with one model is by no means straightforward.

To conclude, false expectations lead to models being created for all the wrong reasons.

# 6 Discussion & Future Work

When we consider our list of general issues, their causes and possible solutions, we must recognize that some of these issues can be difficult to solve, due to numerous practical concerns. Time and money are significant constraints in any real world business projects. To propose that the lack of knowledge of BPM could be solved by placing all stakeholders on a 3 week modeling course is well meant, but unrealistic. Likewise, the lack of adequate tooling support can be leveraged by acquiring professional and feature rich BPM software, but only if such an acquisition fits the budget of the project. Considering the cost of professional BPM suites, small BPM projects are unlikely to invest in such software.

Our research has been of a theoretical nature with most conclusions being based on reason rather than practice. As such, the validity of our research is somewhat indeterminate due its lack of proper grounding in practice. It is for this reason why we have restricted ourselves to describing problems, rather than formulating definite solutions. Although at times we have suggested potential solutions to certain problems, we must emphasize that these should be interpreted as suggestions. Despite the fact that we do not provide any answers to these problems, we do believe that our findings can contribute to those involved in BPM projects. We hope to make the reader aware of certain issues which can arise in BPM projects and why they arise. Though, it is up to the reader how he or she will deal with these issues.

While the topic of BPM has already received considerable attention from the research community, we argue that much remains unquestioned, due to the monotonicity of this research. This is a consequence of the fact that the greater part of this research originates from the computer science discipline. Considering that the 'business management' is part of the BPM acronym, common sense would say that business oriented disciplines such as business administration should play a much more significant part in the advancement of BPM. Ultimately, we believe that it should be these disciplines that determine what information a business process model should describe, whereas the computer science discipline should be focused on delivering the technical support. Moreover, disciplines such as cognition sciences could provide valuable insight into *how* this information should be represented.

Finally, even though BPM is a profoundly technology driven discipline, there still remains much work to be achieved in the development of BPM software, specifically the development of modeling tools. As we have shown in this thesis, principles such as decomposition and views-on-models are instrumental in the representation of complex and information rich models. Modeling tools should assist the user in the application of these principles. However, modeling tools which are currently available on the market only do so modestly. Instead, vendors of these modeling tools tend to emphasize features which have little to do with modeling, such as interoperability with other software solutions.

# 7 Conclusion

In this thesis, we have attempted to assess whether business process modeling can serve as a common language between business and IT stakeholders. We have explored the numerous business process modeling languages and their application, shown the differences between natural language and artificial language and substantiated that system design principles can guide the design of business processes. Moreover, we have revealed a series of issues currently affecting business process modeling activities and exposed their causes based on the theories we've discussed. In the upcoming final paragraphs of this thesis, we will reinforce those results from our research which we deemed most important.

First and foremost, similar to the existence of the great divide between business and IT stakeholders, there exists a great divide between designing business processes and automating business processes. Both activities involve business process modeling, but place fundamentally different demands on what information the model should describe and how it should describe this information. Designing business processes emphasizes stakeholder collaboration and creating a shared understanding of the 'as-is' and 'to-be' processes. The model should embrace these goals by representing the business process in a manner which is comprehensible to all stakeholders. This is best achieved by using a modeling language with a limited vocabulary in combination with text annotation in natural language. On the flip side, automating business processes emphasizes the connection between computer applications and the business process in order to cause computers to exert the desired behavior. The model should embrace this goal by representing the business process in a manner which can be interpreted by a computer. This is best achieved by using a formal modeling language and adhering to the technical constraints imposed by a computer environment.

Secondly, business process management remains an IT-driven discipline and as a result has left its distinct mark on the underlying discipline of business process modeling. The current standard for business process modeling, BPMN, is gradually becoming a more IT-oriented modeling language. This trend makes business process modeling a more powerful technique for IT stakeholders, at the cost of making it less suitable for business stakeholders. Computer programming, formal logic and mathematics are skills, which are uncommon among business stakeholders, but we have shown to be essential in the application of formal modeling techniques. In other words, programming with diagrams rather than text remains programming none the less and does not constitute a viable solution for business stakeholders.

Has business process modeling successfully bridged the business-IT divide? Hardly. But its technology has succeeded in providing a user-friendly programming interface. Just as 'what you see is what you get' interfaces allows computer novices to build websites without knowledge of HTML, CSS and JavaScript, BPM technology might be the first step to the more generic 'what you *want* is what you get' interface, which allows computer novices to get their computer to exert a certain behavior.

In conclusion, we would like to share our view on the importance of modeling in general. While most people see modeling as the process of drawing elaborate diagrams, we see modeling as a structured problem solving strategy and as a means to cope with complexity. Looking back at the advancements the human race has made in the past one hundred years, we observe a trend whereby the complexity of many systems has increased exponentially. Thirty years ago, it was common for a consumer to repair his own washing machine or car. This was made possible due to the limited complexity of these systems, allowing those with a basic understanding of electrotechnics or automotive engineering to perform such repairs. Nowadays, the complexity of these machines has increased manifold and as a consequence, the knowledge required to repair these systems has increased with it. This demand for knowledge has been partially leveraged by improvements made in our educational system and the vast amount of easily accessible information on the Internet.

Unfortunately, there is a limit to the knowledge we can contain, whereas the complexity of a system can be potentially boundless. Thus, we need to invent new and improved ways to handle the increasing complexity of systems and we believe that modeling could play an instrumental part in achieving this goal.

# 8 Literature


[1] C. Pettey, “Gartner Announces Winners of the 2011 BPM Excellence Awards,” 2011. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1588516. [Accessed: 12-Jun-2011].

[2] S. Biazzo, “A critical examination of the business process re-engineering phenomenon,” *International Journal of Operations & Production Management*, vol. 18, no. 9/10, pp. 1000-1016, 1998.

[3] W. Bandara, M. Indulska, S. Chong, and S. Sadiq, “Major issues in business process management: an expert perspective,” *European Council of International Schools*, vol. 2007, pp. 1240-1251, 2007.

[4] M. A. Ould, *Business processes: modelling and analysis for re-engineering and improvement*. John Wiley & Sons, 1995, p. 224.

[5] E. Cardoso, J. P. a Almeida, and G. Guizzardi, “Requirements engineering based on business process models: A case study,” *2009 13th Enterprise Distributed Object Computing Conference Workshops*, pp. 320-327, Sep. 2009.

[6] M. Dumas and W. M. P. van der Aalst, “Process-aware information systems,” 2005.

[7] R. Aguilarsaven, “Business process modelling: Review and framework,” *International Journal of Production Economics*, vol. 90, no. 2, pp. 129-149, Jul. 2004.

[8] H. Mili et al., “Business process modeling languages: Sorting through the alphabet soup,” *ACM Computing Surveys*, vol. 43, no. 1, pp. 1-56, Nov. 2010.

[9] E. Sivaraman and M. Kamath, “On the use of Petri nets for business process modeling,” *Proceeding of the 11th Annual Industrial Engineering*.

[10] R. M. Dijkman and M. Dumas, “Formal semantics and analysis of BPMN process models using Petri nets,” *Queensland University of*, pp. 1-30, 2007.

[11] M. Dumas, A. Grosskopf, T. Hettel, and M. Wynn, “Semantics of Standard Process Models with OR-Joins,” *On The Move*, pp. 41-58, 2007.

[12] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed, “On the suitability of UML 2.0 activity diagrams for business process modelling,” in *Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling-Volume 53*, 2006, vol. 53, pp. 95–104.

[13] K. Phalp and M. Shepperd, “Quantitative analysis of static models of processes,” *Journal of Systems and Software*, vol. 52, no. 2-3, pp. 105-112, Jun. 2000.

[14] G. Abeysinghe and K. Phalp, “Combining process modelling methods,” *Information and Software Technology*, vol. 39, no. 2, pp. 107–124, 1997.

[15] W. M. P. van der Aalst, “Formalization and verification of event-driven process chains,” *Information and Software Technology*, vol. 41, no. 10, pp. 639-650, Jul. 1999.

[16] C. Menzel and R. J. Mayer, “The IDEF family of languages,” *Handbook on architectures of information systems*, pp. 215–249, 2006.

[17] L. Tsironis, K. Anastasiou, and V. Moustakis, “A framework for BPML assessment and improvement: A case study using IDEF0 and eEPC,” *Business Process Management Journal*, vol. 15, no. 3, pp. 430-461, 2009.

[18] J. Recker, “Opportunities and constraints: the current struggle with BPMN,” *Business Process Management Journal*, vol. 16, no. 1, pp. 181-201, 2010.

[19] J. Sauve, C. Bartolini, and A. Moura, “Looking at business through a keyhole,” in *2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops*, 2009, pp. 48-51.

[20] J. Bartenschlager and M. Goeken, “Designing Artifacts of IT Strategy for Achieving Business/IT Alignment,” *Proceedings of Americas Conference on Information Systems, CA, USA*, 2009.

[21] M. Smits, A. Fairchild, P. Ribbers, K. Milis, and E. Geel, “Assessing strategic alignment to improve IT effectiveness,” *BLED 2009 Proceedings*, p. 15, 2009.

[22] T. H. Davenport, *Process innovation: reengineering work through information technology*, vol. 11, no. 1. Harvard Business School Press, 1993, p. 337.

[23] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham, “Impact of service orientation at the business level,” *IBM Systems*, vol. 44, no. 4, pp. 653-668, 2005.

[24] R. E. Grandy, “What Are Models and Why Do We Need Them?,” *Science & Education*, vol. 12, no. 8, pp. 773-777, Nov. 2003.

[25] S. J. B. A. Hoppenbrouwers, H. A. Proper, and T. P. Van Der Weide, “A Fundamental View on the Process of Conceptual Modeling,” in *Proceedings of the 24th International Conference on Conceptual Modeling*, 2005, vol. 3716, pp. 128-143.

[26] Y. Wand and R. Weber, “On the ontological expressiveness of information systems analysis and design grammars,” *Information Systems Journal*, vol. 3, no. 4, pp. 217-237, 1993.

[27] J. Recker, M. Indulska, M. Rosemann, and P. Green, “The ontological deficiencies of process modeling in practice,” *European Journal of Information Systems*, vol. 19, no. 5, pp. 501-525, Jun. 2010.

[28] B. Wyssusek, “On the foundation of the ontological foundation of conceptual modeling grammars: the construction of the Bunge-Wand-Weber ontology,” *Proceedings of PHISE*, 2005.

[29] A. Burton-Jones and P. N. Meso, “Conceptualizing systems for understanding: an empirical test of decomposition principles in object-oriented analysis,” *Information Systems Research*, vol. 17, no. 1, p. 38, Mar. 2006.

[30] G. Kiczales, “Towards a New Model of Abstraction in Software Engineering,” in *IMSA 92 Workshop on Reflection and Metalevel Architectures*, 1992.

[31] E. Ernst, “Separation of Concerns,” *Computer*, 1995.

[32] K. Phalp, “The CAP framework for business process modelling,” *Information and Software Technology*, vol. 40, no. 13, pp. 731-744, Nov. 1998.

[33] K. Rayner, S. J. White, R. L. Johnson, and S. P. Liversedge, “Raeding wrods with jubmled lettres: there is a cost.,” *Psychological science*, vol. 17, no. 3, pp. 192-3, Mar. 2006.

[34] J. E. Hopcroft, R. Motwani, and J. D. Ullman, “Introduction to Automata Theory, Languages, and Computation (3rd Edition),” Addison Wesley, 2006.

[35] G. Hirst, *Semantic Interpretation and the Resolution of Ambiguity*, vol. 34, no. 2. Cambridge University Press, 1987, pp. 131-177.

[36] P. Miller, J. Pane, G. Meter, and S. Vorthmann, “Evolution of Novice Programming Environments: The Structure Editors of Carnegie Mellon University,” *Interactive Learning Environments*, vol. 4, no. 2, pp. 140-158, 1994.

[37] J. Spolsky, “The Law of Leaky Abstractions,” 2002. [Online]. Available: http://www.joelonsoftware.com/articles/LeakyAbstractions.html. [Accessed: 03-Jul-2011].

[38] H. Hanrahan, *Network convergence: services, applications, transport, and operations support*. Chichester, England: John Wiley & Sons Ltd, 2007.

[39] I. Vanderfeesten, J. Cardoso, H. A. Reijers, W. M. P. van der Aalst, and J. Mendling, “Quality metrics for business process models,” *BPM and Workflow*, pp. 179-190, 2007.

[40] V. Gruhn and R. Laue, “Complexity metrics for business process models,” in *9th international conference on business information systems*, 2006, vol. 85, pp. 1–12.

[41] A. S. Guceglioglu and O. Demirors, “Using software quality characteristics to measure business process quality,” *Business Process Management*, pp. 374–379, 2005.

[42] D. Schumm, O. Turetken, and N. Kokash, “Business Process Compliance through Reusable Units,” in *Proceedings of the 1st Workshop on Engineering SOA and*, vol. 6385, no. i, pp. 325–337.

[43] E. W. Dijkstra, “On the role of scientific thought,” in *Selected Writings on Computing A Personal Perspective*, no. 447, E. W. Dijkstra, Ed. Springer-Verlag, 1982, pp. 60-66.

[44] A. Caetano, A. R. Silva, and J. Tribolet, “A method for business process decomposition based on the separation of concerns principle,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 79–85.

[45] T. Elzinga and L. Smiers, “The Common Reference Architecture (CORA) model, Part II,” *via-nova-architectura.org*, no. March, pp. 1-4, 2011.

[46] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, 1997, p. 404.

[47] R. M. Dijkman and S. M. M. Joosten, *Deriving use case diagrams from business process models*. Citeseer, 2002.

[48] N. Barnickel, J. Böttcher, and A. Paschke, “Incorporating semantic bridges into information flow of cross-organizational business process models,” in *Proceedings of the 6th International Conference on Semantic Systems*, 2010, pp. 1–9.

[49] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst, “From BPMN Process Models to BPEL Web Services,” *2006 IEEE International Conference on Web Services (ICWS’06)*, pp. 285-292, 2006.

[50] A. Polyvyanyy, L. García-Bañuelos, and M. Dumas, “Structuring acyclic process models,” *Business Process Management*, pp. 276–293, 2010.

[51] F. W. Taylor, *The principles of scientific management*, vol. 6, no. 1. Harper, 1911, p. 144.

[52] J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuropka, “Configurative Process Modeling – Outlining an Approach to increased Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability,” in *Proceedings of the 15th IRMA International Conference*, 2004.

[53] D. Jagielska, P. Wernick, M. Wood, and S. Bennett, “How natural is natural language?: how well do computer science students write use cases?,” in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 914–924.

[54] M. Pesic and W. M. P. van der Aalst, “A declarative approach for flexible business processes management,” *Business Process Management Workshops*, pp. 169-180, 2006.

[55] G. Redding, M. Dumas, A. H. M. Hofstede, and A. Iordachescu, “A flexible, object-centric approach for business process modelling,” *Service Oriented Computing and Applications*, vol. 4, no. 3, pp. 191-201, Jun. 2010.

[56] J. Gordijn, H. Akkermans, and H. van Vliet, “Business modelling is not process modelling,” *Conceptual Modeling for E-Business and the Web*, pp. 40–51, 2000.

[57] D. Fahland, J. Mendling, H. A. Reijers, B. Weber, M. Weidlich, and S. Zugal, “Declarative versus Imperative Process Modeling Languages: The Issue of Maintainability,” in *Business Process Management Workshops*, 2010, pp. 477–488.

[58] J. Recker, N. Safrudin, and M. Rosemann, “How novices model business processes,” *Business Process Management*, pp. 29–44, 2010.

[59] P. Green and M. Rosemann, “An ontological analysis of integrated process modelling,” in *Advanced Information Systems Engineering*, 1999, pp. 225–240.

[60] J. Becker, M. Rosemann, and C. V. Uthmann, “Guidelines of business process modeling,” *Business Process Management*, pp. 241–262, 2000.