

Rule-based BPM using the AREF modeling language

Master thesis on the why and how of Business Rules in Business Process Management

Jordy Voesten



Colophon

Master Thesis Computer Science
Management & Technology track

Radboud University Nijmegen
Computing Science Department

Radboud University Nijmegen



Capgemini Nederland B.V.



Author: Jordy Voesten
Email: jvoesten@gmail.com
Date: June 18, 2009
Thesis number: IC 607

Supervisors: Dr. Stijn Hoppenbrouwers, ICIS, Radboud University
Prof. dr. Ben Dankbaar, Management & Technology, Radboud University
Patrick Teters, Capgemini

Second reader: Prof. dr. Erik Proper, Capgemini / ICIS, Radboud University

Preface

Lying before you is the culmination of my work as a student of computer science. It reflects my interests in this field by relating the world of IT to its business side and vice versa. With it I finish my official life as a student, starting as an electrical engineering student at TU Eindhoven, switching to computer science at RU Nijmegen, and moving more and more in the direction of information science and the business side of IT, leading me to the management track of the computer science master in Nijmegen and some courses of the software engineering and management master at the IT university / Chalmers in Gothenburg, Sweden.

In writing this final chapter of my student life, this thesis, I have had the help of some people I would like to thank here: first and foremost, dr. Stijn Hoppenbrouwers, my computer science supervisor. Thank you for your excellent supervision, always making the time to talk to me. Even if it was only five minutes, you always found the right ways of nudging me in the right direction or challenging my thoughts. Second, Patrick Teters at Capgemini, for helping me find my way in the Capgemini organization, actively discussing the subject of BPM with me and making me feel appreciated in the work I was doing. Next to them I would like to thank all the other colleagues from Capgemini who helped me with my work, especially Roeland Loggen, Yvette Hoekstra, Michiel Bax and Sander Schoot Uiterkamp and my supervisor from the management track, prof. dr. Ben Dankbaar and second reader prof. dr. Erik Proper.

During the course of my time as a student, a lot of people have contributed to my education and personal development in one way or another and I am grateful to all of them. My time as a student wouldn't have been the same without you!

Last but not least I would like to thank my friends and family, for supporting me in everything I do.

I hope you enjoy reading my thesis.

June 1st, 2009

Jordy Voesten

Abstract

In the world today, agility and compliance are very important factors for running a successful business. Operating globally more and more, businesses are focusing on their cross-functional and cross-organizational processes for their competitive advantage. Business Process Management (BPM) is the discipline working with these processes. This thesis focuses on BPM for processes that can be automated or supported by IT and if (and how) it is possible to improve this by explicitly modeling the Business Rules involved in these processes.

BPM efforts can be divided into two groups: first, a management discipline about discovery, design and deployment of business processes and executive, administrative and supervisory control over them. Second, it is used to describe BPM-enabling technologies: methods, techniques, and software to design, enact, control, and analyze operational processes. These two types of BPM can be spread across the three organizational levels: strategic, tactical and operational, where different parts of a complete BPM approach are important. BPM is involved with different types of processes, that require different support. Important parts of BPM are process models and the process life-cycle.

The business drivers for BPM are making processes within and across an organization more efficient, more effective, more agile and more compliant. Issues are that processes are not agile because process models are interpreted into applications (intermediation), and changing the application takes too long (no progressiveness). Compliance is also partially missed, because transparency of rules in a process is very low. This makes it hard to make people accountable if they do not really understand what they are signing off on.

The Business Rule Approach (BRA) started out as a way of improving communication between business and IT, clarifying the concepts that were discussed. It is *“a methodology—and possibly special technology—by which you capture, challenge, publish, automate, and change rules from a strategic business perspective”* (von Halle, 2001). It is a way of explicitly capturing the rules of the business in non-technical, structured natural language (and/or its formalized counterpart). If properly formalized, rules can be executed by a rule engine and there are some guiding principles like RuleSpeak and SBVR to help define business rules.

This has proven to have a number of important benefits, like creating awareness and transparency about rules and regulations in the organization. This provides true accountability because people understand what they sign off on. Directly executing these business rules in a process has led to very agile process descriptions, where changes are easily made and consequences of that change can be model-checked. Mass differentiation and some knowledge retention are also mentioned as reasons to work with the BRA. Issues with business rules are the black box problem, where users do not trust or are uncertain about the rule system and whether it works as intended. Another issue is the inability of some users to define correct rules and something to watch out for is that rules have to be structured in such a way that the transparency they offer is retained.

The agility problems in BPM, like disintermediation, are solved by empowering business users with understandable rules and the transparency rules provide, helps with the other parts of agility: reusability, progressiveness and integrability. Prerequisite for the success of this combination is the use of executable process models. The BPM issues with compliance are also aided by the

transparency of rules, making users truly accountable, and by the explicit enforcement of rules and regulations.

Business Process Management and the Business Rule Approach are two sides of the same coin: managing the process, both with their own approach for modeling a process. In BPM, modeling is done procedurally, specifying all possible paths. In the BRA, modeling is done declaratively, specifying all applicable rules. In the procedural style, rules are implicit and in the declarative style, flow is implicit. A declarative style gives more freedom in execution, allowing for flexible workflow. Because of this flexibility, not all exceptions have to be modeled at design time, but can be handled during execution of the process. Both styles have their strengths and weaknesses, making them suited for different types of processes.

The AREF modeling language this thesis introduces, supports all types of process modeling from a partially procedural way (as in BPM) to a completely declarative way (as with Business Rules), giving the modeler a choice about the way the process is modeled best. AREF combines these two by making activities the main concept, offering a choice between flow and process constraints (PIC), capturing gateways in rule-based decisions, modeling events and defining state constraints (SIC) and resource rules. A choice between a procedural or declarative model has to be made in AREF. The real-world application of this is shown in a case on the financial world.

The more knowledge work is involved in a process, the more a declarative modeling style is appropriate, because it allows for flexibility, giving the knowledge worker a chance to rely on that knowledge instead of being chained down by the system into a rigid procedure. The process' change intensity, the amount of change the process endures, can be used to tip the scale in one of the directions, where more change favors a declarative style, but each situation requires a new assessment on what is the best approach for a particular process. The modeler background in or experience with logical reasoning could create a need for training in declarative modeling.

Table of Contents

Preface	i
Abstract	ii
Table of Contents	iv
1. Introduction	1
1.1. Research Questions.....	3
1.2. Methodology.....	3
1.3. Scope.....	3
1.4. Reading guide.....	4
2. Business Process Management.....	6
2.1. Definitions: a reference framework.....	6
2.1.1. Business or IT?	6
2.1.2. Levels in the organization	7
2.2. Process types	9
2.3. Process Modeling	10
2.3.1. BPMN	10
2.3.2. Business Process life-cycle	11
2.4. BPM drivers and issues	12
2.4.1. Issues.....	14
2.5. Conclusions	15
3. Business Rules.....	16
3.1. What are they really.....	16
3.1.1. Definitions	16
3.2. The Business Rule Engine.....	18
3.3. Guiding rule creation principles.....	18
3.4. Business Rule Approach	19
3.5. Business Rules drivers and issues	20

3.5.1.	Issues	21
3.6.	Conclusions	22
4.	BPM + BRA = rule-based BPM	23
5.	Process Modeling	26
5.1.	What is a process?	26
5.1.1.	Perspectives in process modeling	26
5.2.	The process modeling spectrum	27
5.2.1.	Procedural/ imperative process modeling.....	27
5.2.2.	Declarative process modeling	27
5.2.3.	Practice: where does it fit?	29
5.3.	Two languages examined.....	30
5.3.1.	BPMN	30
5.3.2.	ConDec	31
5.3.3.	Common elements.....	32
5.4.	Rule-based BPM modeling	33
5.4.1.	Model – View separation	33
5.4.2.	Tokens	34
5.4.3.	Activity	34
5.4.4.	Resource.....	36
5.4.5.	State Integrity Constraint.....	36
5.4.6.	Direct Flow	36
5.4.7.	Process Integrity Constraint.....	37
5.4.8.	Event	38
5.5.	Conclusion.....	40
6.	Choosing the way to go.....	42
7.	Case study: processes in finance and insurance	44
7.1.	The loan sales process	44

7.1.1.	The models.....	45
7.1.2.	Pros and cons	50
7.1.3.	The author's choice.....	51
7.2.	Mortgage application assessment	51
7.2.1.	Pros and cons	52
7.2.2.	The author's choice.....	53
7.3.	Insurance claim handling	53
7.3.1.	The models.....	54
7.3.2.	Pros and cons	55
7.3.3.	The author's choice.....	56
7.4.	Conclusion.....	56
8.	Conclusions and future work	57
8.1.	Questions and answers.....	57
8.2.	Future work.....	59
9.	Bibliography	60
10.	List of abbreviations.....	62

1. Introduction

"The Only Constant In Life Is Change." - Heraclitus of Ephesus, Greek philosopher

If we look at the world around us, we see change everywhere. It is as much a part of life as breathing and eating. We see change in our everyday life, the climate, economy, work, all around us. But we are creatures of habit, change often makes us uneasy. It is however inevitable. The same can be said for enterprises. Most companies don't like change, because for them change means risk and companies don't like risk. But also for them it is true: change is inevitable. This is acknowledged by the fact that agility is a very important keyword in management nowadays. The question is: how is this agility, this adaptability to a changing environment, facilitated?

When looking at the business world today, an increase in the importance of organization-spanning processes and their management as opposed to thinking within some functional silo can be identified. Where functional groups tended to look inside their walls, end-to-end processes have now been identified as a crucial factor for an enterprise, the thing that sets a company apart from its competitors. Gartner's CIO survey ranks Business Process Improvement as a number one priority¹. This shift in focus is mirrored in the flight Business Process Management has taken over the last decades. Processes are hot and everybody has an opinion about them. When looking at the field of Business Process Management (BPM) there is a jungle of different definitions and approaches out there, some more similar than others (Zairi, 1997) (Smith & Fingar, 2003). The approaches can be divided by the level in the organization they reside on: there is the strategic, the tactical and the operational level (Bandara, Indulska, Sadiq, Chong, Rosemann, & Green, 2007), but the most important division is between the supporting IT tools for BPM and BPM as a management discipline.

As a management discipline, BPM's full potential can only be realized when it starts on the strategic level, but most of the BPM tools out there are very much based on the tactical and operational level, connecting information systems². Managing BPM on this tactical level leads to improvement of what is already done, but not to breakthrough competitive advantages. In Figure 1.1: Gartner's BPM hype-cycle

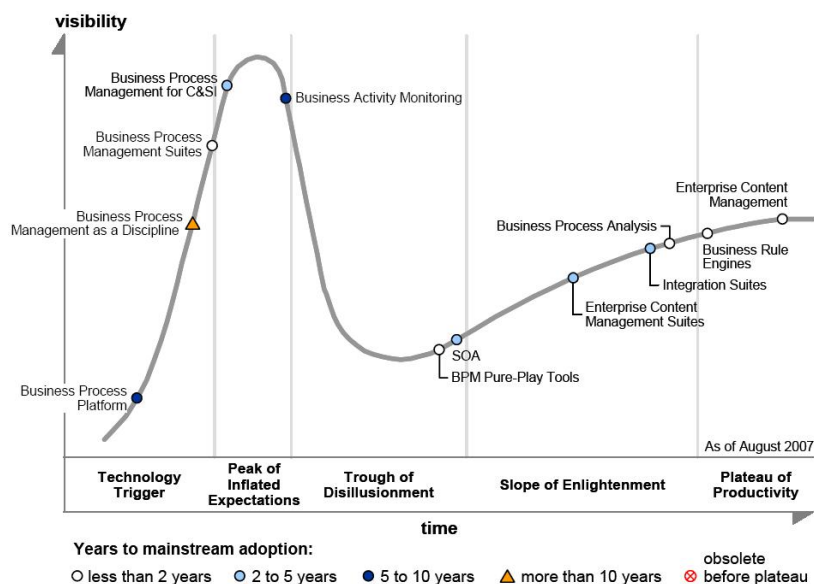


Figure 1.1: Gartner's BPM hype-cycle (August 2007)

¹ Gartner EXP Worldwide Survey of 1,500 CIOs – January 2009 – www.gartner.com

² BPM is Dead, Viva La BPM – Peter Fingar – A BPTrends column – September 2008 – www.bptrends.com

cycle (August 2007) this lack of development is also visible: we see that BPM pure players (companies that have BPM tool manufacturing as a core business) have almost made it through the “Trough” and are beginning to climb the slope. But BPM as a management discipline is estimated just in the first phase of this development and is expected to take more than ten years before becoming common practice: it is slowly starting on the tactical level, but is not effective on the strategic level yet. So BPM, as a complete approach from management to execution, is not yet in the hands of the business, but still very much embedded in tools and information systems. This makes agility hard to accomplish, because the need for change is not apparent on the IT side (the IT department just wants to implement the system’s requirements and change makes that harder). Peter Fingar states that in order to take BPM to the next level a shift in thinking is required, a new paradigm towards an approach focused on *mobile processes*². This means that processes can be changed at run-time, during execution, in order to achieve maximum agility and that processes are owned by the business, not the IT organization. This is also acknowledged in Capgemini’s Technovision document³, introducing *process on the fly*.

Next to the flight BPM has taken in recent years, Business Rules have also gotten a lot of attention. The Business Rule Approach (BRA) by , among others, Ronald Ross argues that when using “*a formal way of managing and automating an organization’s business rules [...], the business behaves and evolves as its leaders intend*” (von Halle, 2001). This means that if the rules that constrain the business are structurally managed, the work will be done as intended. So capturing the rules that are applicable to a certain process will give you a way of structurally manage that process, which makes it very compatible with the BPM approach.

This thesis argues that really fulfilling (part of) the promises made by BPM can be achieved by separating these business rules from the process and making them explicit. Because they are “business talk” (written in non-technical, structured natural they are understandable by the business people and therefore empower the business users. They also enable a new way of looking at process models using a declarative approach to make flexible workflow possible, instead of exhaustively trying to model every possible path. This leaves a lot more freedom to cope with exceptions, without leading to a loss of control (the degree of freedom is determined by the strictness of the rule-based model). In order to find out what the exact business value of a combination between the BRA and BPM (called rule-based BPM) is, both BPM and Business Rules will be explored extensively to find the common ground and where the approaches are complementary.

Since I am studying the management track of the computer science master, I am split between two worlds: Business and Computer science. This thesis has to comply to demands of both fields. Consequently, it is also split between those by focusing on two distinct parts: the first part is mainly about the business value of rule-based BPM, the second part is mainly about formal modeling of a rule-based business process. This part will focus on the meta-models used in both the approaches and looking at combination possibilities between them. The conclusions are again from a more business-oriented perspective. With this I hope to make this thesis interesting for readers from both fields.

³ <http://www.capgemini.com/services/technology-services/technovision/> retrieved on May 19th, 2009

1.1. Research Questions

The questions that will be answered in this thesis are the following:

1. *How can Business Rules contribute to Business Process Management?*

Sub questions here are:

- a. *What is Business Process Management?*
- b. *What are the drivers and issues of Business Process Management?*
- c. *What is the Business Rules Approach?*
- d. *What are the drivers and issues of the Business Rule Approach?*
- e. *Can a rule-based BPM approach help support both the BRA's and BPM's drivers?*

2. *What modeling options are there with rule-based BPM and when should each be used?*

Sub questions here are:

- a. *What is the difference between process modeling in BPM and the BRA?*
- b. *Can their meta-process models be combined?*
- c. *Which approach should be used when?*
- d. *How does it work in the real world? (case study)*

1.2. Methodology

The first question of this thesis will be answered by studying the literature about this subject. A comprehensive overview of the BPM and Business Rules field will be given based on academic as well as business research. During this research domain experts at Capgemini will be consulted to discuss the findings. The second question will partly be based on the literature on declarative modeling, but will mostly focus on the meta-models involved in both BPM process modeling (procedural modeling) as well as process modeling with Business Rules (declarative modeling). Different possibilities of combining these are examined by looking at possibilities in combining their meta-models. Finally a case-study will show how this combination should work in the real world.

1.3. Scope

The field of BPM is a very wide one. A lot is being written about it, especially in combination with the Service Oriented Architecture (SOA) paradigm, but this is not the subject here. The processes studied here are those that can be supported or automated by IT. Next to looking at business benefits for BPM and the BRA separately and together, this thesis will focus on the process modeling aspects of BPM and Business Rules from a more technical point of view. In the discussion about “which modeling approach is best when”, the point of view of a Business Analyst is chosen, to examine the field from a mostly business-oriented perspective. This point of view focuses on the business value of the solution, and not what would be the best solution in a technological sense.

1.4. Reading guide

This thesis is divided into two parts, each answering one of the research questions. In the first part, chapter two and three respectively describe the field of BPM and Business Rules and their business drivers. Chapter four draws conclusions on whether these approaches are suitable to be used together. The second part, starting with chapter five, first looks at process modeling in general, then specifically at two examples, and then proposes a new modeling language for rule-based BPM: AREF (Activities, Rules, Events and Flow). Chapter six contains some advice for making the necessary choices in modeling with AREF and in chapter seven a real-life case is used as an example for the use of AREF. Finally chapter eight draws conclusions on the posed research questions and discusses future work in this area.

PART I

2. Business Process Management

This chapter will discuss Business Process Management by first painting a picture of what is out there in this field, including its history. Then the types of processes involved, the process modeling and the process life-cycle are discussed, followed by BPM's business drivers and the issues it concerns. Finally some conclusions are summarized.

2.1. Definitions: a reference framework

When looking at the field of Business Process Management (BPM) it is quite hard to get an overview of what is out there. When BPM became the next proverbial “silver bullet” in the last decade or so, everyone wanted a piece of the pie. This has led to a lot of different definitions of BPM and a lot of misunderstanding. These definitions exist on different levels and have different purposes. They are not homogenous throughout the field, but the approaches are similar. Two of them are discussed here.

2.1.1. Business or IT?

When BPM is being discussed, whether it is in the academic or the business world, people always mean one of two things: either they are talking about **BPM as a management discipline**, or they are talking about tools, techniques and suites (in other words IT) working with processes: the **BPM-enabling technologies**. The problem is they all call it BPM.

In the management field, Zairi discusses BPM as being “*a structured approach to analyse and continually improve fundamental activities*” (Zairi, 1997). Smith & Fingar, in their book on the third wave of BPM, talk about “*the discovery, design and deployment of business processes, but also the executive, administrative and supervisory control over them to ensure that they remain compliant with business objectives.*” (Smith & Fingar, 2003). According to Gartner⁴ the management discipline BPM has implications on four aspects of the business: Strategy, Governance, Organization and Culture. This includes for instance process ownership. “*... it recommends that organizations shift to process-centric thinking and reduce their reliance on traditional functions and product-centric organizational structures...(in order)...to increase operational performance.*”⁵ This shows that this side of BPM is important for the entire enterprise. It is even argued that BPM doesn't even need technology in order to be a success.⁶ This may be true in theory, but in practice organizations today are too much embedded in their IT to be viewed separately from it, it needs to be used as an enabler.

Thinking about processes in management disciplines goes back a long way. In the context of change management, growing process innovation and radical business process change was captured in the

⁴ “Gartner's Position on Business Process Management, 2006”

⁵ Janelle Hill on BPM as a discipline in “Hype Cycle for Business Process Management, 2007” (Gartner)

⁶ Terry Schurter, “BPM - Practice without Technology”, BPM Group 2005 retrieved from www.it-director.com on January 5th, 2009

term Business Process Re-engineering (BPR) in 1990 by Davenport and Short (Davenport & Short, 1990) and Michael Hammer (Hammer, 1990). While BPR focused mostly on the radical changes in processes in an enterprise, a more incremental change model came from a field called Total Quality Management (TQM), which focused on customer-defined quality in services and products (Reid & Sanders, 2004) and is closely related to the Kaizen (continuous improvement) movement started by the Toyota corporation. A well-known part of TQM is its Plan-Do-Check-Act cycle. These two approaches have led to BPM as a management discipline.

The other side of BPM comprises of the tools and techniques that enable BPM: the BPM-enabling technologies. This is the IT side of BPM involving workflow modeling tools, ERP systems (using best practice processes), but also languages and notations like the Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL). Even Service Oriented Architecture (SOA) is often mentioned in this context. In their discussion of Business Process Management Systems (BPMS), Van der Aalst et al. describe these systems as *“a generic software system that is driven by explicit process designs to enact and manage operational business processes”* (van der Aalst, ter Hofstede, & Weske, 2003). They talk about different types of applications, mainly workflow management systems and define BPM as *“Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information”*. This definition emphasizes the supporting role of systems in BPM.

The technology side of BPM has its roots in Workflow Management (WfM) and can be seen as an extension of this according to van der Aalst et al. (van der Aalst, ter Hofstede, & Weske, 2003). As its name suggests there is a management side to WfM, but it is mainly aimed at modeling workflows. Another influence has been the development in Enterprise Resource Planning systems (ERP). In these systems the *best practices* of certain processes are embedded and when buying an ERP system these best practice processes are part of the deal. ERP is all about applications that support the business. WfM can also be used with applications, when it is a platform for Enterprise Application Integration (EAI). EAI is described as the glue between modular applications and *“allows diverse systems to connect with one another quickly to share data, communications, and processes, alleviating the information silos that plague many businesses”* (Gable, 2002). In this case the workflow model is used to define the communication between the applications and this is made possible through EAI.

So while the business side talks about work processes and their management, the IT side is focused on process models and building suites or (linking) applications based on those processes. This is a different perspective on BPM, built on a different foundation with its own terminology and because these perspectives are much mixed up this leads to a lot of misunderstanding: the *Business-IT divide* in BPM.

2.1.2. Levels in the organization

When looking at an organization it can be divided into three levels: Strategic, Tactical and Operational. This division was linked to BPM by Bandara et al. (Bandara, Indulska, Sadiq, Chong, Rosemann, & Green, 2007) and is a different way to place the current BPM efforts into perspective.

Strategic BPM is the top level. It is aimed at the highest regions of the organization, relating to “*top management support, business and IT alignment, process organization and governance issues*” (Bandara, Indulska, Sadiq, Chong, Rosemann, & Green, 2007). Bandara et al. argue that in order for a BPM effort to succeed top management advocacy is required and “*there should be no gap between organizational strategy and BPM efforts*”, so when making strategic decisions about an enterprise its processes have to be kept in mind and adapted accordingly. Governance of the highest level processes is also part of strategic BPM. Top management should be able to view process metrics and base decisions on them. This level is mainly about the managerial aspects of BPM, with information systems delivering scorecards, dashboards and other metrics as enabling technology.

Tactical BPM is the level below strategic. It is the level of processes where sub goals of the strategy are the purpose of that process. Where the strategy may be to *be the number one BPM software vendor in the world*, a tactic for that can be to *reel in a big contract for selling the software*. This tactical level of BPM is concerned with “*challenges in efforts such as process modeling, process performance measurement and BPM methodologies*” (Bandara, Indulska, Sadiq, Chong, Rosemann, & Green, 2007). It includes for instance Business Activity Monitoring (BAM) and the Business Process Modeling Notation (BPMN) for modeling processes. This is the level of BPM where the application and information system vendors have embraced BPM. Their approach on process centered thinking led to processes being implemented as “*an ad hoc collection of applications, middleware, workflow, manual procedures and user knowledge*”⁷. This means using their existing applications and gluing them together to form the process, the earlier mentioned Enterprise Application Integration (EAI).

Operational BPM is the lowest level on which BPM is practiced. It is involved with the low-level processes that form the most atomic work done in an organization. Since operations is the most easily automated part of an enterprise’s processes, because it changes the least and is the least complex, it is the most IT dependent level. Architecture is very important on this level, deciding how this lower layer is built and communicates to the higher level, which is why the SOA paradigm is often linked to BPM efforts here. Processes on this level are often found embedded in applications. Standardization efforts like the Business Process Execution Language (BPEL) are placed on this level. According to Bandara et al. (Bandara, Indulska, Sadiq, Chong, Rosemann, & Green, 2007) this level relates solely to technical issues in BPM adoption, but managerial choices made on the higher levels have great impact on standardization and architectural issues. Therefore this level also involves some, albeit small, managerial part of BPM, for instance involving process ownership.

When the earlier division between the management perspective and the enabling technologies is superimposed on top of this division in levels, a framework as depicted in Figure 2.1: The complete reference framework can be drawn. This is my view of this field, combining the views of the different perspectives with the different levels. It will be used to place the rest of the research.

⁷ “Gartner's Position on Business Process Management, 2006”

In the framework we can see six possible areas where a BPM effort is working: on the three levels combined with the two perspectives.

2.2. Process types

Next to the different levels on which BPM can be practiced there are also different types of processes BPM has to work with. These are primarily defined by their degree of human involvement and their predictability. They range from structured, highly predictable processes without human involvement (Straight Through Processing, STP) to semi-structured, somewhat predictable processes with a lot of concurrent human involvement and collaboration (Human Interaction Management, HIM). This model was proposed by Roeland Loggen (Loggen, 2008) and is depicted in Figure 2.2: Types of Processes

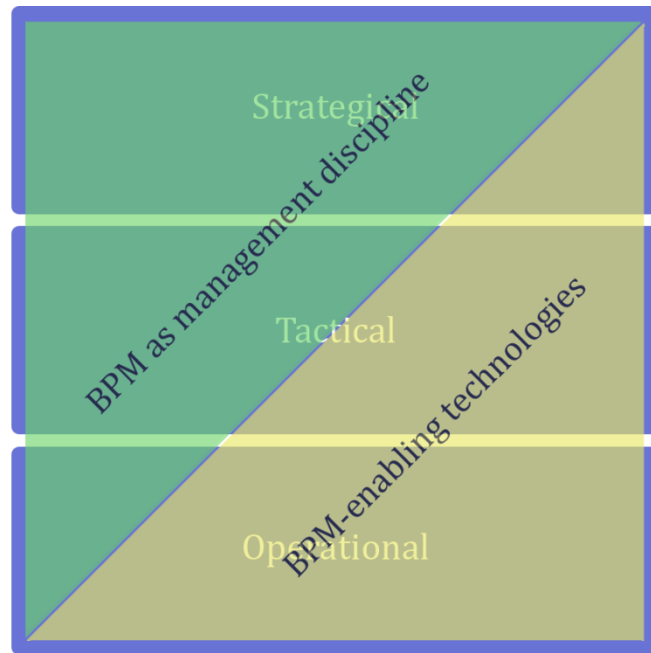


Figure 2.1: The complete reference framework

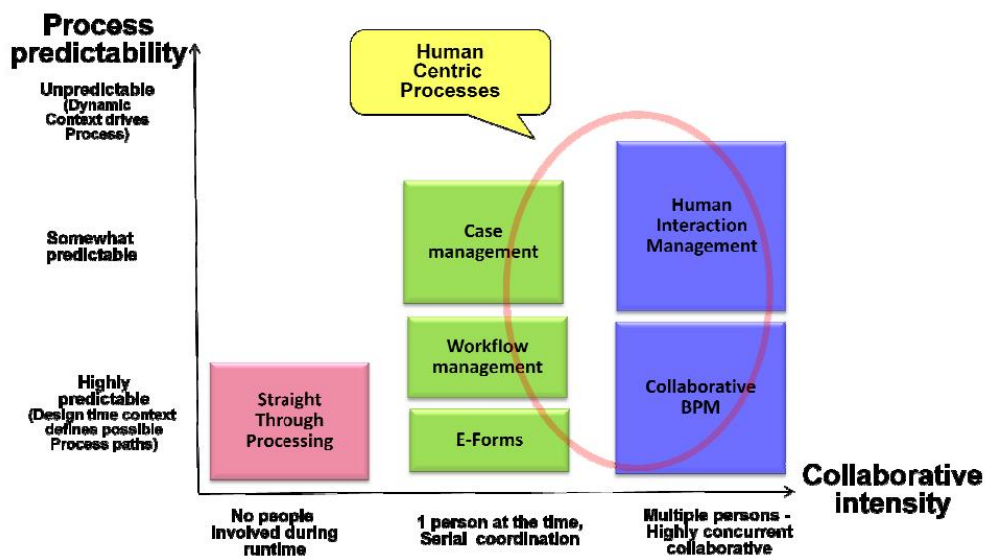


Figure 2.2: Types of Processes (Loggen, 2008)

Straight Through Processing is described as a process where all activities within a process are steered with full automation. **E-forms and Workflow management** is characterized as “pushing (or pulling) work through a serial chain of employees/workstations, each focusing on performing a specific transformation” (Loggen, 2008). In workflow some human interfacing might be required, but these tasks are purely data entry or simple decision tasks. Whether decisions in these processes can be automated (thus making it an STP) depends on the codifiability of the (information needed for the) decision. Case management is involved with more complex decision making procedures. Some

parts of these processes are still predictable, but others need assessment by a case manager, who decides on the (order of) activities needed.

Collaborative BPM and Human Interaction Management (HIM) are fairly different from the others. The fact that more than one person is involved changes the nature of these processes completely. In **Collaborative BPM** *“the key concepts of workflow are still applied, but the key difference is that certain activities within the process require people to collaborate – together research, plan, discuss, negotiate, solve and/or decide, as part of creating value in the process”* (Loggen, 2008). **Human Interaction Management** is primarily about the interactions humans have with one another while working together. It is even more complex because not only are there more people involved, it is also quite unstructured: *“the amount and possible timing of events/signals, possible shifts in goals and possible activities are too much or too unknown to allow modeling the process before actual execution”* (Loggen, 2008). An example mentioned, is the process of discussing something over email: the possible actions are known (reply, reply all, forward, or do nothing), but the order of these and who an email is sent to is not something that can be modeled in a workflow. This distinction in process types will be useful when determining where Business Rules can aid the organization in a BPM effort.

2.3. Process Modeling

Now that it is established what is meant by the different levels and types of BPM, it is time to dig a bit deeper. The common ground between all types and levels is the process model. Without some kind of model that depicts the business process, it is not possible to talk about it on the strategic level, enforce it on the tactical level or enact it on the operational level. A process model is a (graphical) representation of the process and is usually defined as a workflow in BPM efforts. These workflow models are defined in a procedural way, defining steps in a process (activities) and transitions between them (flow).

2.3.1. BPMN

There are multiple techniques available for modeling these processes. On the enabling technologies side and on the tactical level we find BPMN. The Business Process Modeling Notation is one of the standardization efforts to bring the different process modeling approaches together: *“BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance”* (White, 2004). It is a standardized way of modeling a business process. This notation is becoming an increasingly important standard. BPMN is linked to the Business Process Execution Language for Web Services (BPEL4WS or BPEL in short) in such a way that it is possible to generate executable BPEL from BPMN models, making the models executable (operational level, the bottom right corner of the framework). This means that a process can be modeled in BPMN and then translated into BPEL in order to implement the process with web services (in practice this is not completely true yet, because the BPMN syntax contains some ambiguity). Making process models directly executable is quite a new approach in BPM, since normally the models are implemented in a number of linked applications, like an ERP system. An example of a process model in BPMN is depicted in Figure 2.3: A simple BPMN process model.

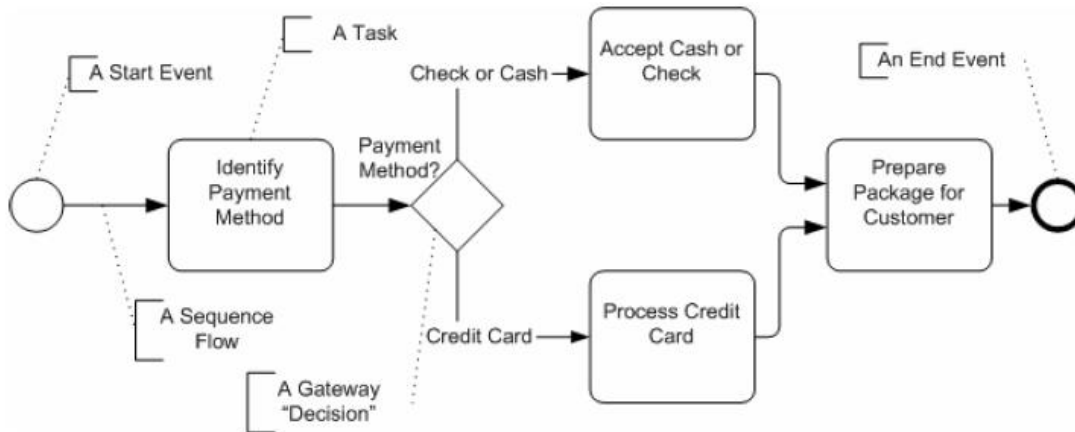


Figure 2.3: A simple BPMN process model (White, 2004)

This is a model of a simple part of a process for getting paid for a package. More on BPMN can be found in section 5.3.1. More detailed information can be found in its specification⁸.

2.3.2. Business Process life-cycle

An important part of BPM is keeping process models up-to-date. When processes change often it is important to record these changes: their life-cycle has to be managed actively. This life-cycle management is an important part of the BPM effort on the management side. Facilitating this Business Process Lifecycle Management (BPLM) is a job for the enabling technologies, but management has to be aware of the work that has to be done. If the life-cycle is not actively managed, changes to the process over time are lost and it is not possible to learn from the past. Metrics concerning this process are important for the strategic level of BPM so enabling technologies have to facilitate the life-cycle model in order to perform BPM on this high level. Figure 2.4 shows an example life-cycle model as used by SAP⁹:

This life-cycle consists of four phases:

- Analysis: current processes, needs and requirements are identified.
- Design: solutions are evaluated, the new processes are designed and modeled.
- Implement: project for the realization, including Go Live
- Monitoring/Run: executing or deploying the process.

This example is based on the traditional improvement lifecycle and is closely related to the Plan-Do-Check-Act cycle by Deming¹⁰, used in TQM. This approach is also described by



Figure 2.4 SAP process management life-cycle

⁸ BPMN specification 1.1 by OMG, <http://www.bpmn.org/Documents/BPMN%201-1%20Specification.pdf> retrieved on May 5th, 2009

⁹ SAP website: <https://www.sdn.sap.com/irj/bpx/bpx-cycle>

Georgakopoulos and Tsalgatidou in their paper on BPLM (Georgakopoulos & Tsalgatidou, 1998)

For life-cycle management to work, it is necessary for the models of the processes that are automated to be linked to the actual execution of the process. If this is not done the changes will be made in the applications in an ad hoc matter and the model will not have changed accordingly. This will lead to a mismatch between the business view of the process (the model) and the IT view (the applications), and life-cycle management will no longer be possible. Also some kind of process change history or versioning helps with managing the process life-cycle, giving it traceability. These and other issues will be discussed in the next section on drivers for and issues with BPM.

2.4. BPM drivers and issues

The development of BPM has had two groups of drivers, like the two perspectives it came from: market-pull and technology-push. The market-pull drivers are the main reason BPM exists. They surfaced at enterprises that realized that a process-centric view of the organization is required in order to gain competitive advantage. These drivers are located in the top left corner of the framework (Figure 2.1): How can business processes on a strategic and tactical level be improved to help the enterprise. The Business Process Improvement (BPI) methodology was introduced for this. Its goals are (Zairi, 1997):

- Making processes more effective (producing the desired results).
- Making processes more efficient (minimizing the resources used).
- Making processes adaptable (being able to meet changing customer and business needs).

Making processes able to meet changing needs is also called *agility*. This agility can be necessary on multiple levels. It can be needed for processes inside the organization, but in recent years of mergers, takeovers and collaboration it has become more and more business critical to align a company's processes with its suppliers, customers or its new mother company. In the *Workflow Handbook 2003*, Faget et al. discuss this needed agility: "*Agility in businesses is nothing new, but with the acceleration of technology and business cycles, it is becoming extremely important for business to remain viable*" (Faget, Marin, Mégard, Owens, & Tarin, 2003). In their discussion on evolving information systems in an agile and reactive manner, four characteristics are mentioned:

- **Disintermediation:** empowering experts, process owners and decision makers to directly define the new rules, processes or applications without intervention of the IT department.
- **Integrability:** the need to preserve the independence of the existing systems, with respect to new systems.
- **Progressiveness:** the ability to be modifiable in order to meet the rapid evolution of the organizational environment.
- **Reusability:** the ability of sharing best practices within the organization, thereby increasing its reactivity and efficiency

Another driver often mentioned in the context of BPM but not mentioned in BPI is *regulatory compliance*. The reason it is not mentioned might be because it is more about policies, rules and regulations than it is about processes. In most BPM efforts, these are embedded in the process

¹⁰ Plan-Do-Check-Act: <http://en.wikipedia.org/wiki/PDCA>, retrieved on June 12th, 2009

models, but it has been often argued that these need to be separated (Faget, Marin, Mégard, Owens, & Tarin, 2003). However, compliance is still used as an important reason for starting a BPM effort. Compliance is for instance to the Sarbanes-Oxley Act¹¹ is mentioned by Gartner, Global 360¹² and BEA systems¹³ (now part of Oracle) as an important BPM driver. It consists of four distinct parts¹⁴:

- **Defined processes** – Use well defined processes.
- **Ways to enforce rules and regulations** – Ensure that regulations are being met across the board.
- **Accountability** – Assign specific parts of the process to specific people or groups.
- **Audit Trail** – Reporting capabilities that create an audit trail that can be easily accessed and provided to regulators when necessary.

These four drivers, also acknowledged by Goedertier et al. (Goedertier, Haesen, & Vanthienen, 2007), are summarized in Table 2.1: Drivers for BPM (market-pull)

Drivers for Business Process Management (market-pull)	
Effectiveness	Producing the desired result
Efficiency	Minimizing resources used
Agility	Disintermediation Integrability Progressiveness Reusability
Compliance	Defined processes Ways to enforce rules and regulations Accountability Audit Trail

Table 2.1: Drivers for BPM (market-pull)

Technology-push is another type of driver for the current BPM offering. Because ERP and other system vendors wanted to embrace BPM they tried to rephrase their offerings to match the BPM demand. This was done without the necessary paradigm shift as described by Smith and Fingar (Smith & Fingar, 2003). Normally technology-push is based on developing a certain technology and showing customers the benefit and therefore selling it. In this case however, technology-push was

¹¹ http://en.wikipedia.org/wiki/Sarbanes-Oxley_Act

¹² "BPM for compliance to reduce risk, time and cost" - www.global360.com

¹³ "BEAParticipate: BPM for Compliance" by Sandy Kemsley at www.column2.com

¹⁴ "Beyond Sarbanes-Oxley *The Benefits of BPM for Compliance*" – www.upsideresearch.com

used to sell enterprises advanced WfM systems and integrated applications based on processes under the name of BPM, promising the market-pull drivers discussed above. Making processes more efficient and more effective is definitely possible with these systems, and that explains their successes in BPM so far. Also compliance can partly be accomplished, but agility is a big problem with these types of systems. That brings us to the issues in BPM.

2.4.1. Issues

The biggest issue with BPM at the moment is the lack of agility. Process models are designed, but after that they are given to IT department to build applications for them or to link this process together with existing applications. This linking of applications (Enterprise Application Integration) is a tight coupling mechanism. At the moment of design the process model is changeable, after that it is embedded in the applications that are linked together to perform the process. A process is designed or re-designed in a BPR effort and then cast in cement by handing it over to the IT department. Changes to these processes require new application development and this is expensive and takes too long. Agility is lost once the process moves beyond the design phase because there is no disintermediation, progressiveness or reusability. What is missing is the flexibility in the enabling technology on the tactical and operational level so that it can cope with the volatile nature of the enterprise of today. This flexibility cannot be achieved with the current workflow and application technology as a basis. Some BPM vendors have acknowledged this and with their new products it is possible to directly execute the process models that are designed, but this is a very recent development and very few organizations are using them yet. Another option is to execute the process model through services that are defined on a layer between the models and the applications, using the Service Oriented Architecture, but this is also not common practice yet.

There is an issue with the process models as well: rules and policies are implicitly embedded in these models, so they are not visible on the outside. Without explicit rules it is very hard to see what level of compliance there actually is. The fact that when shown together, these models form a big pile of spaghetti doesn't make it transparent enough to see the consequences they have for compliance. So while the processes are defined, can the process owner really be held accountable for the results they produce? I don't think so.

A third problem, described by Smith & Fingar (Smith & Fingar, 2003) as *the effect of time*, states that a business requirement goes through multiple iterations until it reaches a technical specification. In these iterations technical details of changes in code may have an unclear effect on the business consequences. Interpretation is part of every iteration, creating a chain of Chinese whispers¹⁵, leading to errors because the business no longer 'owns' the process. There is no disintermediation. This is a common problem in all Business-IT communication, so also in BPM with its two sides.

¹⁵ Chinese whispers: a line of people is formed and the first person whispers a message in the second person's ear. The second whispers it into the third person's ear and so on. The message delivered to the last person will not be the same as the one the first person started with. "Errors typically accumulate in the retellings, so the statement announced by the last player differs significantly." http://en.wikipedia.org/wiki/Chinese_whispers, retrieved on June 12th, 2009

2.5. Conclusions

This chapter has delved into the world that is BPM, trying to give an overview of what it contains. Within BPM, two important sides were identified: BPM as a management discipline and BPM-enabling technologies. These can both be practiced on the three different organizational levels: Strategic, Tactical and Operational, although not equally divided amongst those. Different types of processes involved in BPM were identified followed by a short description of process modeling and the importance of process life-cycle management. The most important part of this chapter looked at the drivers for BPM, which are effectiveness, efficiency, agility and compliance, and the issues connected to those drivers in current BPM offerings: BPM implementations through building or linking applications is not agile and compliance lacks the necessary transparency.

3. Business Rules

This chapter will start by explaining what Business Rules are. Subsequently, Business Rule Engines, the RuleSpeak guidelines, the Business Rules Approach (BRA) including some of its history and drivers for the use of Business Rules will be discussed, followed by some of the issues involved. The chapter will end with some conclusions.

3.1. What are they really

‘Defining Business Rules - What are they really?’ was the title of the paper resulting from the GUIDE project at the end of the nineties (Hay & Healy, 2000). It was one of the first to thoroughly describe business rules. The project team for this project included big names in this industry, like Terry Halpin (ORM modeling), Ronald Ross and Barbara von Halle (currently seen as *the* Business Rule experts) and John Zachman (the Zachman architecture framework). Business Rules can simply be seen as the rules of the business: *“A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The business rules which concern the project are atomic: that is, they cannot be broken down further”* (Hay & Healy, 2000). An example of a rule is: ‘Every invoice has to be paid within 30 days’. So what is so revolutionary about this? Businesses have always functioned according to rules, policies and regulations. The difference here is making them explicit. Usually these rules are somewhere buried in the code of a system, or in the head of a business analyst, for instance when (s)he is designing process models in BPM. Making these rules explicit, developing a discipline for the discovery and management of these rules is what the Business Rules Approach (BRA) is about. This principle has been described earlier as *“Separating the know from the flow”* (Burlton, 2001). The most important part is how they are expressed: declaratively. That means declaring what rules are governing the process, not how the process works when governed by the rules. An example of this in the process for renting a car: the process can contain an activity that describes how the age of a client is checked step by step. In rules, a rule saying *“a customer must be 18 or over”* would cover the same part of the process, without giving a detailed description of how this should be done. This leaves more freedom during the execution of the process. How this works exactly is explained in the chapter on process modeling in section 5.2.

3.1.1. Definitions

Like in BPM, not everyone agrees on the definitions used. This depends mostly on the point of view chosen. In the original GUIDE project (Hay & Healy, 2000), an Information System point of view was chosen, focusing on fact models and data. This thesis will not follow their definition completely because the thesis is written from a more business oriented perspective, and is concerned with activities, events and roles in the organization. Taveter and Wagner have looked into the different types of Business Rules described in the literature and this has led to the following distinction (Taveter & Wagner, 2001): **Structural assertions** are the basis on which the rules are built. They are the terms and facts used to define the ontology of the rules, the description of the world, the structure of the enterprise. Terms are basic definitions of entities and facts relate terms together. An example of a term would be *Employee*, defined as someone who works for the company and *Parking*

space as a location at the company where a car can be parked. A fact relating these terms together can then be: *'Employee has Parking space'*. These can be base facts or derived facts. The latter are a result of the later mentioned derivation rules, whereas the former has to be defined at design-time. These structural assertions are not always included in listings with types of business rules. They are often described as the *business vocabulary, ontology or fact model*.

If the structural assertions do not count as a type, **Integrity constraints** are the first type of Business Rules. They are assertions that must be satisfied and come in two forms: first are state integrity constraints, which must hold at every point in time (invariants), like 'Customer must be at least 18 years old' or that may be temporarily violated as long as they are satisfied before the end of the process, like "The application must be printed and signed by the client". Second are process integrity constraints, which restrict admissible transitions between states, like flow in a process model, defining which activity can follow after the previous one. The difference between flow and process constraints is that the latter limit possibilities, telling us where the process *cannot* to go, whereas the former defines the only possible path, where the process *can* go. This difference will be highlighted again later on when talking about procedural versus declarative modeling in section 5.2

Second are **Derivation rules**. These are either a mathematical calculation or an inference. They do not need to be stored separately, because they are derived by an inference mechanism. Inference is something that human beings do all the time. When for instance two rules (instances of structural assertions) exist: 'All cars have four wheels' and 'A Ford Mustang is a car' then we derive that 'A Ford Mustang has four wheels' without that being an explicit rule. This is an example of a derived fact mentioned earlier. Other derivation rules (often called production rules) have an 'If -> Then' format. The inference mechanisms used for this are described in section 3.2.

Third are **Reaction rules**, responses to events, also called event-action rules or ECA rules for Event-Condition-Action. They state the condition under which action must be taken: triggering event conditions, pre- or post-conditions. They relate these rules to agents: *"Reaction rules define the behaviour of an agent in response to environment events (perceived by the agent), and to communication events (created by communication acts of other agents)."* (Taveter & Wagner, 2001). Agents can be anything from a person to groups or the entire organization. An example of this is the rule "A client is rejected and the police is notified, if fraud is detected" where *fraud detected* is the event that requires reaction and the fraud detection system is the agent.

A final type of Business Rules mentioned by some are the **Authorizations**. These rules are about the power, rights and duties of agents in the organization. This is about assigning work to certain roles, giving permissions or obliging an agent to do something like 'Only the people manager can recommend an employee for a promotion'

3.2. The Business Rule Engine

When reading about a business Rule solution a Business Rule Engine is often mentioned. Business Rules has its earliest roots in expert systems and this is where the inference mechanisms mentioned earlier come from. A Business Rule Engine (BRE) is based on this principle of inference and is used to derive new information from the current data and the state of the process at hand. There are two types of inference: Backward and forward chaining¹⁶. The difference between these is that in forward chaining you start with the

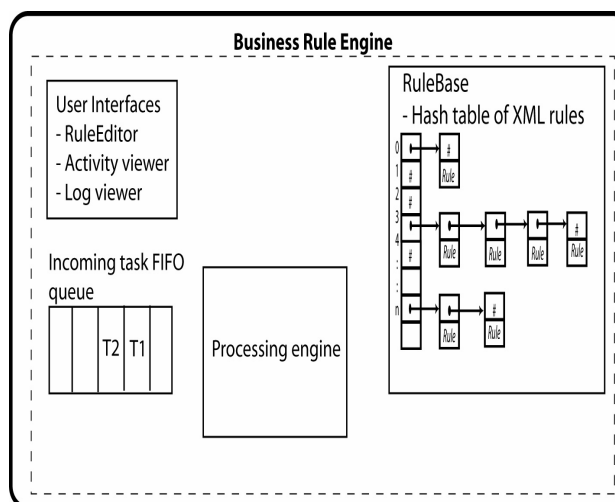


Figure 3.1: Anatomy of a BRE

data at hand, working from there with the rules (find an If clause that matches your data) in order to reach a goal. A goal in business rules could be to check if a customer is a loyal customer (in order to give him or her a discount). The rule-base is searched for rules that can be fired (executed) and the system works its way forward from that. A widely used algorithm for this is the Rete algorithm, because simply starting with the data available could get out of control easily when more than one rule can be executed. Rete is a smart way to avoid this explosion of possibilities. Backward chaining works the other way around, starting with the goal and working back from there to see if the required data for that goal is available. A BRE is an engine that implements backward or forward chaining or both, in order to apply business rules. It could be used to make decisions inside a process. The decision to give a certain customer a discount could be based on the following business rules:

- If a customer is a loyal customer, (s)he gets a 10% discount.
- If a customer spends more than € 10,000 a year and is a customer for more than 3 years, then (s)he is a loyal customer.

The engine could look at the rules and the data and decide whether a certain customer deserves a discount or not. An example of a rule engine is depicted in Figure 3.1: Anatomy of a BRE (Dimitoglou, 2004), where the queue contains rules that can be fired (executed), which are selected from the RuleBase (which contains all rules) by the processing engine, according to the algorithm that is chosen.

3.3. Guiding rule creation principles

As the examples show, business rules are written in natural language. Because natural language is often ambiguous in its semantics, it is difficult to formalize and a formal way to define rules is necessary in order to base IT systems on them. To overcome these issues, a number of guiding rule creation principles have been defined by the Business Rule Group¹⁷ or its members. The first one

¹⁶ http://en.wikipedia.org/wiki/Backward_chaining and http://en.wikipedia.org/wiki/Forward_chaining

¹⁷ www.businessrulesgroup.org, home website of the Business Rule Group

was RuleSpeak, created by Ronald Ross. RuleSpeak is *“a set of guidelines for expressing business rules in concise, business-friendly fashion. It is not a language or syntax per se, but rather a set of best practices for speakers of English”*¹⁸. It is meant to help business users define well-formed rules, in order to prevent ambiguity. Next was the Business Rule Manifesto¹⁹, describing *The Principles of Rule Independence*, a document containing overall guidelines for working with rules. The most recent milestone in formalizing rules is the Semantics for Business Vocabulary and Business Rules (SBVR), an OMG standard for *“documenting the semantics of business vocabularies, business facts, and business rules”*²⁰ These guiding principles can all be used together.

3.4. Business Rule Approach

Like in BPM, Business Rule Management (BRM) is more than just technology. Next to the tools required to use business rules, it is also a methodology, a discipline for the discovery and management of rules. The Business Rule Group has developed a way of doing this simply called the Business Rule Approach (BRA). Barbara von Halle describes it as: *“A business rules approach is a methodology—and possibly special technology—by which you capture, challenge, publish, automate, and change rules from a strategic business perspective”* (von Halle, 2001). When fully implemented, this leads to a Business Rule Management System/Suite (BRMS). A BRMS is more than just the use of a rule engine, but also includes rule (life-cycle) management. Next to making it possible to define rules in the system, it is also able to execute them. It offers more than just modeling, making it the complete package for using business rules in the enterprise. This approach to rules was important from the start of the movement because it focused at often changing rules and policies. It goes back to the mid-nineties, when it was used to try and retain knowledge in an organization when people left. *“Business rule solution: A systematic way of capturing, documenting and retaining the business rules prevents the loss of knowledge when people leave”* (Ross, 2003).

Next to this business perspective and the expert system roots we saw, business rules have some other roots as well. Database management and Object Orientation also led to rule-based systems. Figure 3.2: History of Business Rules (Dimitoglou, 2004) shows the four sources of rule research. The database management systems (DBMS) were concerned with integrity constraints, derivations and reaction rules (respectively assertion, view and trigger) and Expert systems' interest was in the inference/derivation rules. These both were data-oriented approaches. Object Oriented systems focused on integrity constraints and derivation (OCL and Prolog) and the Business Rules Approach to System Development brings all these together from a business perspective. These last two are process-oriented approaches to rules. As you can see Business Rules has quite a history, so why has it become such a hot item in recent years? Because its business drivers have become increasingly important.

¹⁸ Definition of RuleSpeak, retrieved from <http://www.rulespeak.com/> on May 24th, 2009

¹⁹ <http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf>

²⁰ Semantics of Business Vocabulary and Business Rules (SBVR), v1.0 retrieved from www.omg.org on May 24th, 2009

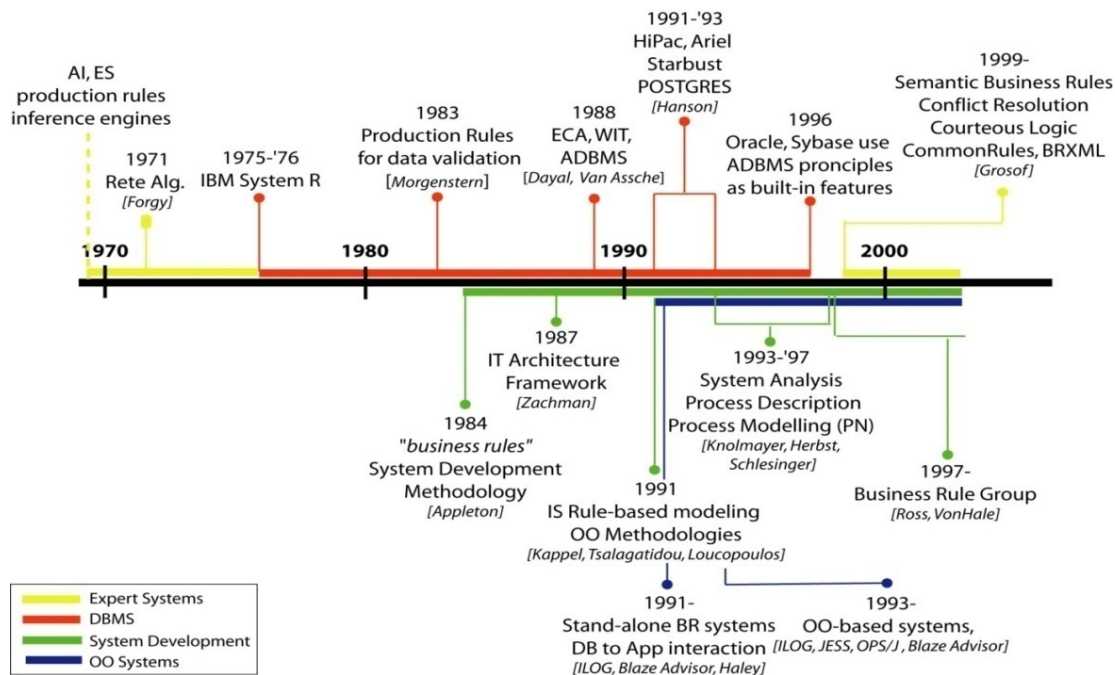


Figure 3.2: History of Business Rules (Dimitoglou, 2004)

3.5. Business Rules drivers and issues

The Business Rule Approach mentions a number of reasons for using business rules, and a number of problems it helps to solve. The biggest two reasons mentioned are agility and compliance. As discussed in the previous chapter, agility can be broken down into four parts: disintermediation, integrability, progressiveness and reusability. Barbara von Halle (2001) writes that the BRA makes change less disruptive and costly (progressiveness) and helps systems be delivered faster and cheaper, because of rule reusability. And that it brings business people closer to the system specification (disintermediation). She also says that “A business rules approach is a formal way of managing and automating an organization’s business rules so that the business behaves and evolves as its leaders intend.”, which will lead to compliance because business users can now really be held accountable (if the rules are properly organized, making their use transparent). Third thing she mentions is that because business rules are now documented and accessible people in the organization can find the rules. This visibility leads to more awareness in the organization as well as being able to relate rule changes to outcome changes instantly.

Ross mentions six problems that the BRA solves (Ross, 2003):

- Ad hoc rules are no longer made up as the process goes along.
- Miscommunication of key business concepts is avoided by defining them in facts and terms.
- Inaccessible rules are taken out of the source code and put into the hands of the business.
- Mass differentiation is no longer a problem because for each customer type different rules can be applied.
- The need to keep up to speed is facilitated by the easily changed rules
- (Some) knowledge no longer leaves the company when the people who have it leave, because it is captured in the rules.

Chisholm adds a last promise to the BRA (Chisholm, 2003): Engineering compliance (different from the earlier mentioned compliance, which is regulatory compliance) means that there is no interpretation in the steps between business requirements and code, and thus avoiding the “Chinese whisper” problem mentioned in the previous chapter (section 2.4.1). This is tightly related to the disintermediation part of agility.

Drivers for Business Rule Management	
Agility	Disintermediation Progressiveness Reusability
Compliance	Ways to enforce rules and regulations Accountability Engineering compliance
Rule awareness	Rules are known throughout the company
Mass differentiation	Different rules for different customers
Knowledge retention	Keep knowledge from leaving the organization with people that have it

Table 3.1: Drivers for BRM

3.5.1. Issues

As with BPM there are some issues with the Business Rule Approach. The most important one is the *black box problem* with Business Rule Engines, as described in (Chisholm, 2003). This problem arises because users of the system cannot see inside the engine how it works and therefore do not trust it because of uncertainty or feeling a loss of control. When this happens people start a shadow set of spreadsheets which they use to check the outcomes of the engine. This extra work leads to a loss in production. Another problem is that the users want to know how the rules work, being unsure if the rules they describe have the desired effect. So this issue is mainly about uncertainty and trust issues with this new way of working, because when errors occur, they are almost always due to users having written a rule incorrectly. So an issue related with this is the inability of users to produce correct rules.

Another possible pitfall is the overwhelming amount of rules that an organization has to work with. Rules have to be structured in such a way that the transparency they offer is retained. Otherwise accountability and rule awareness become hard goals to reach.

3.6. Conclusions

This chapter explained what Business Rules are and what different types can be distinguished. Some guiding rule creation principles, inference and Business Rule Engines were discussed, as well as the Business Rule Approach. Its history was described, relating it to Expert systems, DBMS, System development and Object Orientation. This was followed by the drivers for and issues in the BRA, where agility, compliance and awareness were widely seen as the most important reasons to work with Business Rules. Trust and uncertainty issues with the technology and some users' inability to produce correct rules were recognized as issues, and keeping a good overview of the rules is a prerequisite for keeping the offered transparency.

4. BPM + BRA = rule-based BPM

Chapters two and three contained the drivers and issues for BPM as well as the BRA. This chapter takes a look at how these two can work together forming rule-based BPM.

Since agility and compliance are the two overlapping fields of interest, the focus is on these. Table 4.1: Fulfillment of business drivers by BPM and BRA contains an overview of all drivers found, with an indication whether either of the fields fulfills it. (~ meaning partially)

Driver	Fulfilled by BPM	Fulfilled by BRA
Effectiveness	Yes	Yes
Efficiency	Yes	Yes
Agility	No	Yes
Disintermediation	No	Yes
Integrability	No	~
Progressiveness	No	Yes
Reusability	No	Yes
Compliance	~	Yes
Defined processes	Yes	Yes
Ways to enforce rules and regulations	~	Yes
Accountability	~	Yes
Audit trail	Yes	Yes
Engineering compliance	No	Yes
Rule awareness	No	yes
Mass differentiation	No	yes
Knowledge retention	~	yes

Table 4.1: Fulfillment of business drivers by BPM and BRA

As can be seen in the overview, the Business Rule Approach is a very good addition to BPM. There is one important consideration to keep in mind though: the issues with BPM are partly connected to the current way of implementing it, so there are more ways to solve its problems. When looking at

the components of agility, it shows that disintermediation (empowering the business) does not work in BPM. This is partly because process models are interpreted into systems and the business is no longer in control. Making process models directly executable would solve this, if the business users completely understand how each part of the process model works. This is easier in rules because they are very clear in their meaning and can be (should be) defined by the business. Integrability, progressiveness and reusability are things that can be helped by building BPM on a Service Oriented Architecture, but this does not exclude business rules from the equation. Better yet, the transparency that comes from using business rules helps to see necessary changes and makes it more clear which parts can be reused as a service, helping with reusability and providing true progressiveness.

As far as compliance is concerned, BPM already helps to achieve this for a part. Obviously defined processes and audit trails are possible when the process is completely captured in a process model (if the IT department interpreted the model correctly). Rules and regulations can often be captured in this process model, but are interpreted into a system and remain invisible to the outside viewer, so checking this compliance is hard work. This is even worse for accountability, because when business users are not empowered, how can they be held accountable for their processes? Should they be accountable because they sign off on something they cannot really see the consequences of? Using business rules here clearly makes enforcing rules and regulations possible more strictly because of engineering compliance and in a much more transparent way because of the use of business language (if structured in such a way that an overview is still possible). It also makes business users really accountable, because now they really understand what they are responsible for. And even though with only business rules the process itself might at first be less transparent (as was visible with the black box problem), an audit trail can always be followed after execution of a process has taken place (provided all that data is stored somewhere).

So in order to make BPM more compliant and especially more agile, disintermediation is the most important goal to reach. Using explicit business rules is a very big step in the right direction, giving business users a direct link to the execution of their processes. It helps narrow the gap between the two sides of BPM, management and supporting IT. Next to that it delivers the added bonus of rule awareness, some knowledge retention and the possibility for mass differentiation, making rule-based BPM the method of the future.

PART II

5. Process Modeling

This chapter will examine the possibilities for modeling a process. It will look at the different perspectives that can be used to look at a process model. Subsequently a process modeling spectrum will be drawn up with on the one side the modeling of processes according to most workflow and BPM tools: procedural process modeling and on the other side process modeling with rules: declarative process modeling. In the range of places in this spectrum of modeling approaches, two positions on the far ends will be elaborated on further for an in depth analysis of the concepts used in these approaches. Finally a combined modeling language and its meta-model will be introduced.

As discussed in the previous chapter, executable process models solve a lot of the current issues in BPM. For the remainder of this thesis an executable process model is the goal when discussing process and rules modeling. This means the model must span both the tactical as well as the operational level of BPM-supporting IT.

5.1. What is a process?

Before discussing process modeling, it needs to be clear on what is meant by “a process”. It is important to note the difference between process and procedure. Historically, processes in BPM have been described as a linked list of activities, specifically describing what steps have to be taken, one after the other. This is actually a procedure, which is more strict, a recipe on how to do something. A process does not necessarily have to be defined so strictly. When looking back on how a process went, the steps and their order can of course be written down as a recipe, but this cannot always be done beforehand. Chapter two showed the complex and collaborative processes in Human Interaction Management, like the “simple” process of discussing something using email. The steps in this process are clear, but the procedure for who is going to reply or forward what, when and to whom, is not. A procedure is the option closest to our mindset when thinking about a process, but there is also another way to approach this: in a declarative way. This way the steps are defined but there is more flexibility in the transitions between these steps. How this works is shown in section 5.2.2

5.1.1. Perspectives in process modeling

When looking at process modeling, there are multiple perspectives that can be used to look at a model. Three of them are shortly discussed here: the control-flow perspective, the resource perspective and the data perspective. The **control-flow perspective** “defines in which order activities can be executed”, the **resource perspective** “defines which (human) resources are authorized to execute each of the activities and how the actual resources are allocated to execute the activities” and the **data perspective** “defines which data elements are available in the process and how users can access them while executing activities” (Pesic, 2008). Choosing one of these perspectives as the most important one in a model, defines what a model looks like. This becomes important when the different views of the model are discussed in section 5.4.1

5.2. The process modeling spectrum

The two ends of this spectrum represent two different paradigms in process modeling: procedural (also called imperative) modeling versus declarative modeling. The battle between these two paradigms is not a new one. It has been fought in the world of programming languages where languages like C, Fortran, Cobol and most recently Java (Object Oriented, but still imperative in style) on one side were representing the imperative approach and Prolog is the most well-known language representing the declarative approach to writing a computer program. This clash was arguably won by imperative languages since they are used predominantly in practice, but the discussion has never really ceased (for example (Lau & Vanden Bossche, 2002)) and has been sparked again concerning process modeling. A big difference between the two is the possible amount of flexibility it gives the execution of the model. This is explained while discussing the two ends separately. The spectrum describes possible combinations of procedural and declarative modeling, where some parts of the process are modeled procedurally and some declaratively. The bigger the influence of one or the other, the more towards one of the ends of the spectrum that position is located. The two ends in extreme are:

- purely procedural, only activities and flow, no rules (there are implicit rules, but not modeled).
- purely declarative, only activities and rules, no flow (there is implicit flow, but not modeled).

5.2.1. Procedural/ imperative process modeling

The procedural way of describing a process is what BPM and Workflow is currently all about. A step-by-step description of what should be done to reach the process goal: a procedure for running a business process. This is the most natural (or maybe just habitual?) way for people to think about processes and therefore widely used. This is often mentioned as the *how* of process modeling, because it gives a detailed description of how the process should be run. Rules and regulations concerning the process are often modeled implicitly in such a model. Transitions between activities are described as flow, and branching in the flow is done by using gateways for decisions.

The problem with this is that all possible paths in this step plan have to be modeled and this is often difficult because it is hard to think of all possibilities exhaustively at design time. Another reason this could prove hard to do is because of the spaghetti such a model can become, when lots of branching possibilities occur in a process model. The implicit modeling of the rules also leads to some intermediation and accountability issues, as mentioned in chapter two.

5.2.2. Declarative process modeling

Another way of modeling a process is declaratively. This means declaring *what* needs to be done in the process (the activities or tasks) and declaring which rules or constraints the model has to comply to. Declaring something about your process is exactly what Business Rules do. A declarative process model will exist of activities in the process and constraints concerning these activities. This is often explained as defining the *what* and not the *how* of a process, but there is another way of looking at this. It is all about constraining possibilities, which is saying more than purely *what* should be done. What should be done can be seen as the set of activities or tasks in your process and the data used (the structural assertions). If there are no constraints on these activities and they can be executed in random order, they define *what*. The moment you start adding control-flow constraints on transitions between activities (be it optional or mandatory) you leave the *what* and start specifying

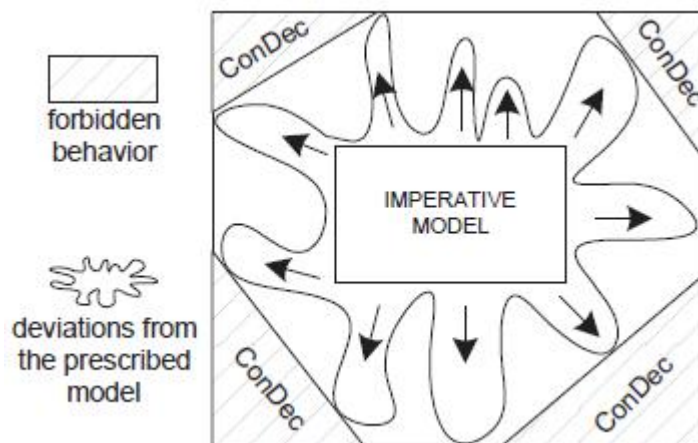


Figure 5.1: Declarative ConDec vs. Imperative languages (Pesic & van der Aalst, 2006)

the *how*. The more constrained a model becomes, the more is defined about *how* the process works, making it less flexible. The ultimately constrained model is as strict as a procedural model, allowing no freedom to the user executing it. This means that the declarative rule approach gives you the option to be very flexible and ad-hoc, but can also be as inflexible as with a procedural model, giving you a choice. This freedom can be very useful in process modeling and prevents over-specified procedural models. This happens when something cannot be modeled procedurally in a natural way (like mutually exclusive activities mentioned later in this section). The downside is, it makes the choice in modeling options more complex to grasp. How this works is shown in Figure 5.1: Declarative ConDec vs. Imperative languages is compared to imperative languages. This figure clearly depicts the difference in approach between a declarative and procedural approach. Where the procedural (called imperative here) defines all possible paths in the center rectangle, the declarative constraints only restrict the parts that are forbidden. This leaves room for deviations from the prescribed model, giving the model more flexibility (Pesic & van der Aalst, 2006)

These control-flow constraints between activities can be seen as pre and post conditions for an activity and these are process integrity constraints. When looking at all four types of business rules, we can see that the integrity, reaction rules and a part of derivation rules are related to the control-flow perspective, while derivation rules about facts and authorizations are aimed at the data and resource perspective. Data and resource rules form triggers that are represented in the control-flow perspective as events.

What are the benefits of declarative control-flow? The ordering of activities becomes more flexible, it is more up to the user who knows what (s)he is doing and does not need a rigid procedure. So even though it sounds like a contradiction, the use of rules leads to flexibility. Another benefit is that some constraints, like mutual exclusivity between activities can be modeled more naturally with constraints instead of procedure (Pesic & van der Aalst, 2006). For example, if someone wants to model mutual exclusivity with a procedure, (S)he defines two activities and makes a decision gateway before it that branches into two paths leading to the activities. First of all, the decision would have to be made *before* executing either of the activities, which is premature if the execution of the activity does not follow immediately after it. Second, when one of the activities would later be replaced by one that is not mutually exclusive with the other activity but the person changing the model does not know the rationale behind the decision gateway, (s)he will leave the gateway there, even though these two activities need no choice between them. Declaring that the two activities can never be executed in the same process instance is the more natural way of making sure they are mutually exclusive.

5.2.3. Practice: where does it fit?

In industry, Business Process Modeling approaches are procedural at the moment and some Business Rule tools have made a start with declarative reasoning, but in academia truly declarative modeling has been thoroughly researched in recent years. Figure 5.2 places the current practices in the spectrum from procedural to declarative and adds the new AREF language. All languages depicted here can be used for building executable models, as mentioned earlier. Only BPMN has some problems here, having a slightly ambiguous execution syntax, which would lead to problems of interpretation (Wohed, van der Aalst, Dumas, ter Hofstede, & Russell, 2006).

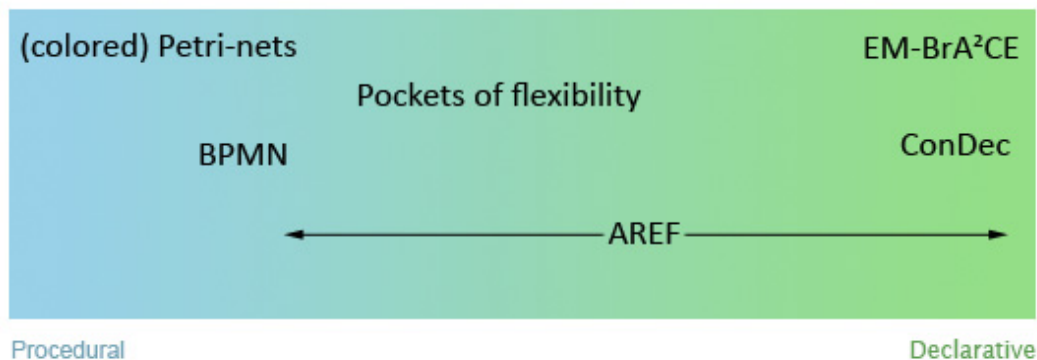


Figure 5.2: Modeling approaches in the spectrum

(colored) Petri-nets are a well-known procedural form for designing process models. It is probably one of the oldest ways to model a process as it was invented in 1939 by Carl Adam Petri. It describes places, transitions and directed arcs. Petri-nets have an exact mathematical definition and execution semantics²¹. Petri-nets are strictly procedural.

BPMN is the OMG's (Object Management Group) modeling standard for business process models. It is an attempt to unify all BPM tools under one standard. It is basically procedural, but it allows for the definition of some ad-hoc activities, which can be executed in random order. This makes it slightly supportive of declarative modeling. More on BPMN can be found in section 5.3.1.

Pockets of flexibility as proposed by Sadiq et al. (Sadiq, Sadiq, & Orłowska, 2001) describes making distinct parts of the process model more flexible by loosely or partially specifying the model and making the true specification at runtime, unique for every instance. So high-level processes are procedural, with some sub-processes defined more declaratively.

The **EM-BrA²CE framework** as developed by Goedertier et al. (Goedertier, Haesen, & Vanthienen, 2007) is one of the declarative approaches. It contains rules for control-flow, data and organization and sees the process model as "State space + Transition constraints". The complete framework consists of *Policies and regulations, business rules, activities, events, agents and roles and business concepts*.

The **ConDec** language (and the related DecSerFlow language), together with its modeling tool DECLARE by Pesic et al. (Pesic & van der Aalst, 2006) form another declarative approach. It has

²¹ http://en.wikipedia.org/wiki/Petri_net retrieved on May 4th, 2009

focused on the control-flow perspective, but does mention the data and resource perspective. Key concepts are *activities* and *constraints*. ConDec itself is purely declarative, but in combination with YAWL (Yet Another Workflow Language) it can be part of a higher level procedural model (Pestic & van der Aalst, 2006). More on ConDec can be found in section 5.3.2.

AREF is the modeling language proposed in this thesis for rule-based process modeling and spans a large part of the spectrum because it contains choices on the use of a more procedural or more declarative style. It is discussed in section 5.4

5.3. Two languages examined

5.3.1. BPMN

As mentioned before, BPMN is OMG's modeling standard for process modeling. It is rapidly becoming the industry standard. It is a procedural notation, but the OMG has tried to incorporate some flexibility in the form of ad-hoc sub-processes. Modeling constructs in BPMN are *Flow objects*, *Connecting objects*, *Swimlanes* and *Artifacts*. This last category is purely meant to provide some extra information about a process, but it has no execution purposes, so is not very interesting at this point. Swimlanes are used to separate resources and come in two forms: *Pools* and *Lanes* within a pool. These pools and lanes can be people or departments within an organization or other organizations. The difference is that communication between pools is done with connecting objects: message flow. Within a pool (and in or between lanes), activities and events (both flow objects) are connected with sequence flow (connecting object). To split and join sequence flow BPMN uses gateways (flow object). There are multiple events and gateways available to form all sorts of workflow patterns, but a basic model consists of activities connected with sequence flow in a procedural way. An activity can be of the type sub-process, containing more steps. In here ad-hoc activities can be placed if required.

The wsper²² community has published a meta-model of BPMN (Figure 5.3: BPMN 1.0 meta-model by wsper.org), based on the OMG documentation and even though it is version 1.0 this is a good summary of how the concepts in BPMN are related together. The BPMN specification²³ contains more details on the available concepts, but this is outside the scope of this thesis. For now it is enough to know that the main concepts are *activities* and *events* connected by *flow* (which can be split and joined by use of gateways).

²² <http://www.wsper.org> retrieved on May 5th, 2009

²³ BPMN specification 1.1 by OMG, <http://www.bpmn.org/Documents/BPMN%201-1%20Specification.pdf> retrieved on May 5th, 2009

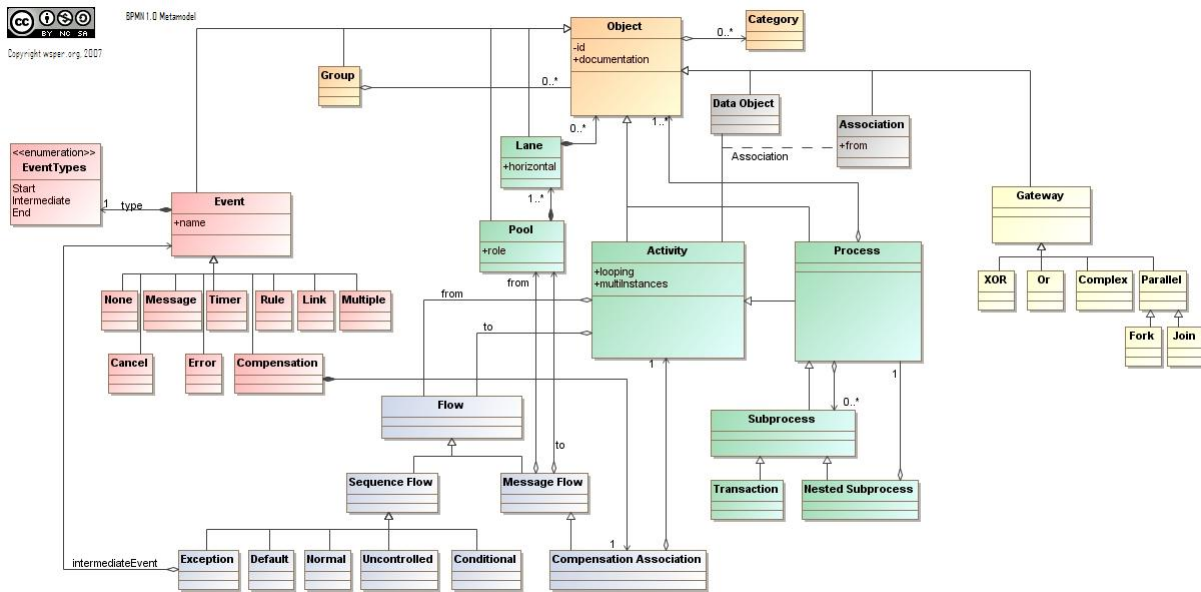


Figure 5.3: BPMN 1.0 meta-model by wspan.org

5.3.2. ConDec

The ConDec language is all about defining relationships between activities. It does so by using “an open set of constraint templates” (Pesic, Schonenberg, Sidorova, & van der Aalst, 2007). The semantics for these relationships are based on Linear Temporal Logic (LTL)²⁴. As the “T” suggests, this form of logic contains a timing aspect, which is very useful for constraining relationships between activities. The basics of logic variables and connectives will not be discussed here, but the temporal aspects are interesting to look at here. There are three unary operators (working on one variable): Next, Globally and Finally, and two binary operators (working with two variables): Until and Release. Table 5.1 shows the symbols used for these operators and the meaning they have.

Name	Symbol	Meaning
Next	$\circ A$	A has to hold at the next state (point in time)
Globally	$\square A$	A has to hold at every state in the subsequent path
Finally	$\diamond A$	A has to hold somewhere in the subsequent path
Until	$A \cup B$	B holds at the current or future position, A has to hold until that position.
Release	$A \mathcal{R} B$	B holds until the first time A is true, or forever if A is never true

Table 5.1: LTL temporal operators, based on http://en.wikipedia.org/wiki/Linear_temporal_logic

These operators are used to define the templates ConDec uses, but additions can be made freely. An example of a template is the *response* template. In LTL this looks like: $\square (A \rightarrow \diamond B)$, meaning that for the entire model (globally \square) it holds that if A happens, then eventually (finally \diamond) B will hold. If A and B are activities, that means that activity B will eventually be executed if A is executed. These

²⁴ http://en.wikipedia.org/wiki/Linear_temporal_logic retrieved at May 5th, 2009

templates return in the language proposed in this thesis and will receive more attention there. An example of a ConDec model in DECLARE (the prototype implementing ConDec) can be seen in Figure 5.4: ConDec model for fractures treatment where a number of activities can be seen, constrained by a number of business rules in writing as well as graphically. In this model a doctor always starts with examination, but after that (s)he has the freedom to choose from a number of activities. There is no end point in this process, the doctor closes an instance when (s)he is done (if all constraints are satisfied).

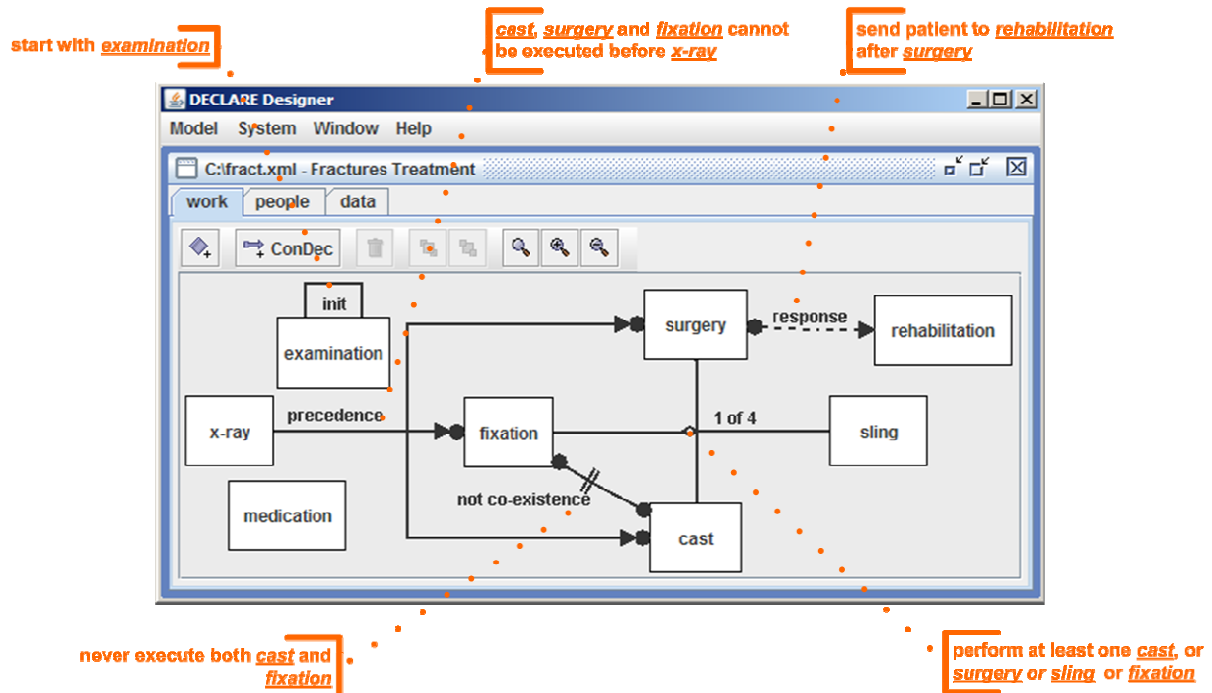


Figure 5.4: ConDec model for fractures treatment (Pesic M. , 2008)

For every ConDec model an automaton or state machine can be deduced with the possible activities as arcs that bring the machine from one state to the next. This opens up a number of possibilities for model checking and verification (Pesic, Schonenberg, Sidorova, & van der Aalst, 2007), making sure the process model is correct.

When executing a ConDec model, users see a list of activities that can be started (the work list). This list depends on the constraints on these activities and is dynamic. The execution of one of these activities triggers new constraints which lead to a changed list of activities to choose from next. In ConDec, the control-flow perspective is the most interesting because of its new approach. Resource and data perspectives are not directly visible in this view (but are available in DECLARE) and are therefore left out of this overview, making *activities* and *constraints* its most important artifacts.

5.3.3. Common elements

The big difference between the BPMN and ConDec is the way control-flow is done, but there are a lot of common elements in both procedural and declarative process modeling. The most obvious one is *Activities*, grouping together work to be done. Another one is the *Data model* the process is built on. This is a requirement for an executable process model, procedural or declarative. *Events* occur in all processes, but are modeled differently. In procedural languages like BPMN they are

modeled in the (sequence) flow steering a process in a different direction depending on the event. In declarative languages the event is modeled as the condition for which a certain reaction is appropriate (reaction rule). In both, events are always triggered by data or a human interaction. Human interaction is constrained by resource assignments, defining which user has which rights for executing activities. Resource assignment can also be used for defining which resources can be altered in which activity, for example with certain documents as resources.

In defining the options for control-flow, a choice has to be made. This is reflected in the model proposed in the next section.

5.4. Rule-based BPM modeling

In this section a modeling language is proposed and its meta-model is discussed. It is a graphical notation that combines elements of modeling constructs for processes and rules. It is called AREF, which stands for Activities, Rules, Events and Flow, the four main elements of the language. The purpose of the modeling language proposed, is to make it possible to model both the process as well as the business rules that the process has to comply to. For that reason it spans a big part of the modeling spectrum (Figure 5.2). Because the choice for a position in the spectrum depends on certain aspects (more on this in the next chapter) the model reflects the choice modelers have to emphasize the more structured procedural side of the process or the more flexible declarative side. In other words the modeler has the possibility to choose a position in the spectrum, based on the needs for a specific process.

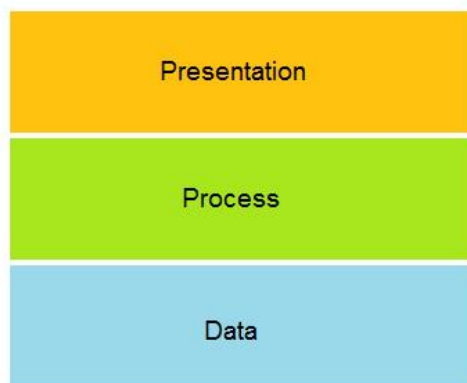


Figure 5.5: Three-tier architecture for rule-based BPM

The model has a three-tier architecture, meaning it consists of three layers (Figure 5.5: Three-tier architecture for rule-based BPM). These layers are important during design as well as during execution of a process model. First is the presentation layer. At design time, this layer contains the editor in which the models can be built supporting different views on the model (more on this in the next section). During execution time, this layer contains the screens users interact with, as well as an overview of the current instance of the process, again possible in different views. Then there is the

process layer, containing the model of the activities, flow and/or rules. Last is the data layer. The data layer has to be created first in order to be able to define Business Rules. It is however possible to model activities and flow or process constraints first, in order to start with an overview of the process. The meta-model for AREF is depicted in Figure 5.6: Meta-model for rule-based process modeling in AREF

5.4.1. Model – View separation

Before the meta-model for AREF can be discussed something has to be explained about the difference between a model and its view(s). This is closely related to the earlier discussed perspectives. BPM models in general are focused on control-flow. Activities and transitions between them are the central interest of the modeler and therefore central to the picture that is drawn. If we use a different perspective to look at a model, for instance the resource perspective, we could draw

a picture with figures for different people (or groups) in the organization and draw lines between these figures and the activities they connect to, making the resources the most important and central part of the picture. This does not change the underlying model, only the way we look at it: its view. This separation becomes important when a choice has to be made to model flow or constraints in the model.

5.4.2. Tokens

In order to explain some of the differences in execution semantics of the model, the concept of tokens has to be introduced. Tokens are also used in BPMN (and before that in Petri-nets) and show which activity is currently in the *started* state, where the locus of execution is currently at. Activities which are at that time in the *active* state are eligible to receive this token when an activity is finished or an event takes place. All activities in this active state are placed in a *work list*. This work list is the list of activities a user can choose to execute. Which activities are available in the work list depends on the rules on, or flow between activities. So at the start of a process, a token is created with the instance of that process and this token is passed along from activity to activity. It can be compared to a ballgame where only the person with the ball can do something.

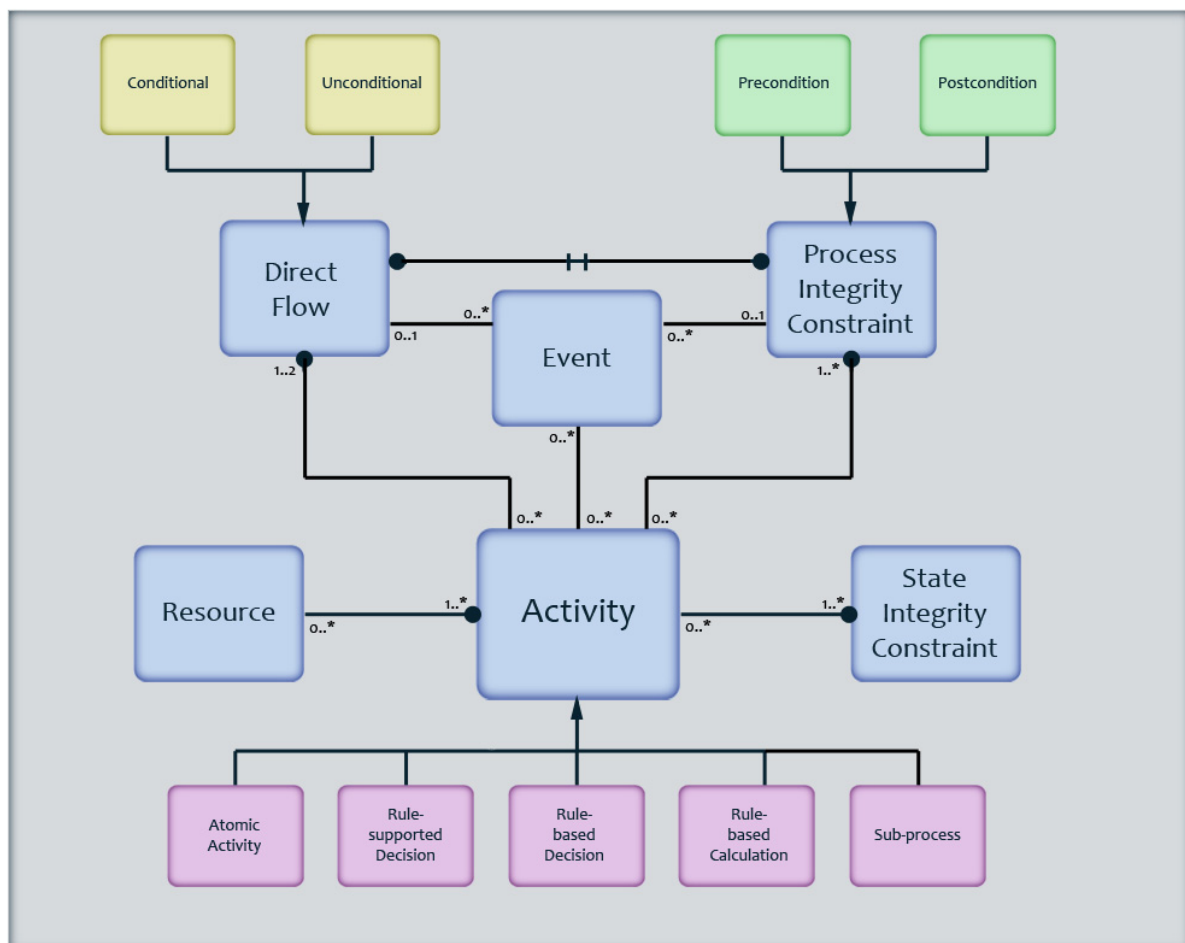


Figure 5.6: Meta-model for rule-based process modeling in AREF

5.4.3. Activity

The central point of the meta-model is the **Activity**. This is the most important part of every process model and all other parts of the model are related to this. There are five types of activities possible.

The **atomic activity** is generic activity that is used for anything other than a decision or calculation and describes an atomic task or work item. Decisions come in two types, **rule-based** and **rule-supported decisions**. These have Business Rules (derivations) as the ground for a decision, but the difference is that in a rule-based decision the system decides, whereas a rule-supported decision is always made by a **resource**, in this case a person or group within the company. Then there is the **rule-based calculation**, for instance to calculate (i.e. derive) a discount for loyal customers. Although decisions and calculations are both based on derivation rules the difference is that a calculation does not contain a gateway and can therefore only have unconditional outgoing flow. Last type of activity is the **sub-process**. This type contains a sub-process in a collapsed state. It is used as a grouping mechanism, but it can have its own integrity constraints. All activities contain a list of rules that are linked to that activity. This can be integrity rules as well as resource rules. Events also contain such a list, containing the involved reaction rules.

Activities have a *state* they are in. Possible states are *inactive* (when the activity is not in the work list), *active* (when the activity is in the work list), *started* (when the activity is currently being executed), *finished* (when the activity has been executed) or one of the *exception* states, being *timed out*, *cancelled*, *expired* or another state that the modeler needs. The exception states are always the result of an event. Once a token arrives at an activity, it remains inside an activity from the started until the finished state except for when an event occurs, which may copy, jump or reroute the token (see section 5.4.8 Event)



Figure 5.7 Atomic activities

The basic form of an activity in the model is a rounded rectangle with the activity's name on it. For all activities except the sub-process, this rectangle also contains a figure showing the resource this activity has been assigned to.



Figure 5.8 Rule-based/supported activities

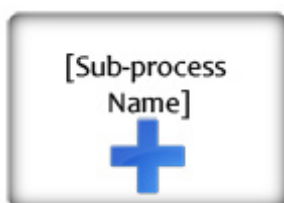


Figure 5.9 Sub-processes

The other activities contain markers depicting their type: a calculator for calculations, a legal hammer placed on a diamond (gateway) shape for rule-supported decisions, a legal scale placed on a diamond (gateway) shape for a rule-based decision and a plus-sign for a sub-process.

When an activity is selected the list of all related rules is shown. These rules can be resource rules or integrity constraints.

5.4.4. Resource

An activity must be assigned to a **Resource**. This can be a person or a group and if no authorizations are defined this automatically becomes the group “All”, containing all users. Only exception to this are the **sub-process activities**, which are not assigned to a specific resource. Other resources that can be linked to activities are documents. Resource rules on documents define during which activity this document is involved in the process. These rules are about rights management.

The resource that the activity is assigned to is depicted on the activity by the color of the figure on it. This figure depicts a person or group. Resource rules about documents are not depicted, only shown as rules in the list within the activity.

5.4.5. State Integrity Constraint

With each activity come integrity constraints. The constraints that apply during the execution of an activity are **State Integrity Constraints (SICs)**. At the end of the execution of an activity, all its state integrity constraints must be *true* (or consciously overruled by a human actor), during execution they may be violated temporarily. So these are rules that apply to a specific activity, but they can be reused for other activities. All SICs must relate to at least one activity. All rules relating to a decision are also modeled as SICs. These rules are shown in the rule list of an activity. There are a lot of different possibilities for SICs. Examples are “*Name, Date_of_birth, Address, Nationality and Client_nr must be filled in.*” and “*The application must be printed and signed by the client.*”

5.4.6. Direct Flow

How activities relate together is either defined as Direct Flow or as Process Integrity Constraints, depending on the chosen view. Using both in one view would make the model very difficult to comprehend. This is because similar signs are used to depict them in the model, but their meaning is very different (flow defines what the next step *can* be, constraints define what the next step *cannot* be). They can however be present in the same model, since direct flow is a form of a very strict constraint. It can be difficult to comprehend because direct flow not only contains the constraint that “activity B has to follow activity A”, but also that “no other activity than B can follow activity A”. This temporarily blocks all other activities that are in the active state. Other Process Integrity Constraints are often not that strict. So this is where the modeler chooses whether (s)he wants to take a procedural or declarative approach, because the choice of the view implicitly defines the level of strictness in the constraints. This choice can be made on all levels of the model, so a sub-process can be declarative while the higher level process is procedural. But if both constraints and flow are possible in the same model, why the choice? Because the flexibility but also the complexity of using constraints has to be a deliberate choice for your process. Direct flow can be modeled with constraints (although this is not easily depicted graphically), but most constraints cannot be modeled with direct flow, providing the declarative approach with a more expressive toolbox.



Figure 5.10: Direct Flow

Direct flow is the same as sequence flow in BPMN: a procedural way of going from one activity to another, like a one-way street. It is depicted as a direct arrow from one activity to another, defining an ordering between these activities. It can be either **conditional** or **unconditional**, based on the type of activity it is connected to. Conditional direct flow can only come from a decision activity, where the outcome of the decision defines if the condition for this flow is met. All direct flow not coming from decision activities is unconditional. Direct flow must connect to at least

one and at most two activities, but multiple flows with a shared starting or end point can be joined

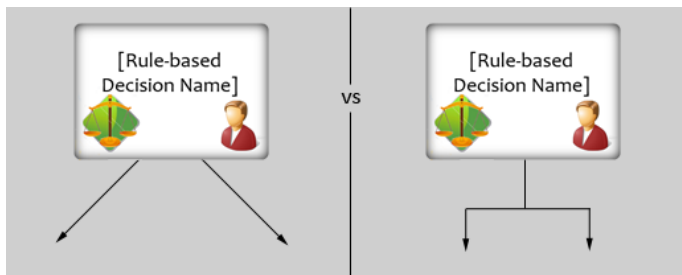


Figure 5.11: separate vs. graphically joined Direct flow

graphically to keep the model as uncluttered as possible, without changing its meaning (Figure 5.11). Direct flow is drawn with an arrow. The arrow is the same for conditional as well as unconditional since conditional flow can only come out of decision activities. Direct flow connected to an event can only be outgoing.

5.4.7. Process Integrity Constraint

Process Integrity Constraints (PICs) are the declarative way of relating activities together. This is done in the same way as in ConDec, using graphical depictions of LTL formulas. PICs are either **pre** or **post conditions** for an activity or both a post condition for one and a pre condition for another activity. PICs must connect to one or more activities. The difference between state and process integrity constraints lies in the timing. SICs are examined during an activity and PICs before or after an activity. PICs are purely about control-flow, only constraining the possible transitions from one activity to the next, pre or post an activity. The fact that SICs have to evaluate to true at the end of an activity can also be seen as a post condition, but it is actually a requirement for an activity to go into the *finished state*. PIC post conditions are only evaluated after an activity has reached that finished state or before a new activity has gone into a *started state*.

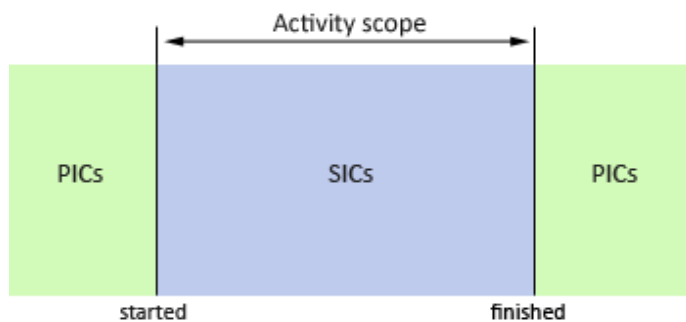


Figure 5.12 PIC and SIC scope

As with ConDec, these process integrity constraints are based on LTL expressions, and have a graphic representation in the model. This is done to make it as easy as possible for non-experts to work with them. Seven examples are given here, but more constraint patterns can easily be added. A special case of a PIC is a conditional PIC, which can only be connected to decision activities. These are dotted versions of a PIC to indicate that these are related to conditions. This distinction is intended to help keep the model overview clear, because unconditional PICs are always valid and conditional PICs are not. The *init* and *grouping* constraint cannot be conditional.

As mentioned before, it is hard to graphically depict a constraint modeling direct flow, because this temporarily blocks other activities. An option to overcome this is grouping together activities that have to follow each other directly. For example when activity B has to follow activity A, as far as all relating constraints concern they apply before (pre condition) the start of A and after (post

condition) the end of B. This is only allowed for single, non-splitting direct flow, otherwise this should be modeled as a sub-process.




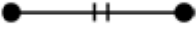
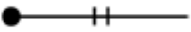
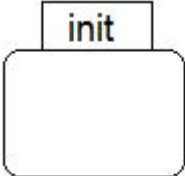

Graphic symbol	Meaning	LTL expression
	The activity on the left has to eventually be followed by the one on the right. (<i>response</i>)	$\Box (A \rightarrow \Diamond B)$
	The activity on the right has to be preceded (somewhere in time) by the one on the left. (<i>precedence</i>)	$\Box (\Diamond B \rightarrow (\neg B \cup A))$
	Combination of both previous constraints. (<i>succession</i>)	$\text{response} \wedge \text{precedence}$
	These two activities are mutually exclusive. Once one is executed the other may not be. (<i>not co-existence</i>)	$\neg (\Diamond A \wedge \Diamond B)$
	If the activity on the left is executed, the activity on the right may not be. (<i>responded absence</i>)	$\Box (A \rightarrow \Box \neg B)$
	Pre condition for all other activities: this activity needs to be executed first (<i>init</i>)	$\forall x (x \in \text{Activities} / \{A\}) (A_precedence_x)$
	Group together, direct flow modeled as a constraint (<i>grouping</i>)	$\Box (A \rightarrow \circ B)$

Table 5.2 Standard Process Integrity Constraint patterns

5.4.8. Event

Events are tricky things when modeling a process. An event can be defined as anything that happens, making them hard to capture. In this model, events are seen as data triggers: a changed data field that is in the scope of a reaction rule. This data field can be set by a system, for instance a BAM (Business Activity Monitoring²⁵) or CEP (Complex Event Processing²⁶) platform as was suggested

²⁵ http://en.wikipedia.org/wiki/Business_activity_monitoring, retrieved on 21-04-2009

²⁶ http://en.wikipedia.org/wiki/Complex_Event_Processing, retrieved on 21-04-2009

in (von Ammon, Emmersberger, Springer, & Wolff, 2008) or a human actor (for instance when an alarm button is pushed).

Since events are such a tricky part of the model a number of possible scenarios have been identified. These scenarios are combinations of three sorts of possibilities: activity-based or process-wide events, events that require immediate action or eventually need follow-up and events in the view of Direct Flow or with Process Integrity Constraints. These lead to eight possible combinations for

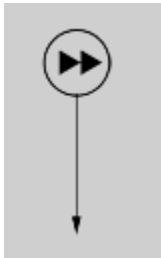


Figure 5.13: Immediate process-wide event with Direct Flow

which the modeling options are summed up in Table 5.3: Event modeling possibilities. Activity-based events are always modeled on the border of the activity they belong to, process-wide events are always floating in the model. Both types have no incoming direct flow or pre condition PICs. The difference between them is that process-wide events can always be triggered, while activity-based events can only be triggered during the execution of that activity.

Events are modeled as circles containing a single triangle mark (“play”) when eventually response is required and a double triangle mark (“fast forward”) for events with immediate response (figures 5.13 and 5.14). Like activities, events have a list of related rules. An example rule for an event is “Immediately go to *Fill out application* if the *application expiration date* expires.” In which the keyword “immediately” in the rule definition signals that the event requires immediate response.



Figure 5.14: eventually, activity-based event with PIC

The difference between “immediate” or “eventually” is that the current running activity at the time of the event behaves differently. For an “immediate” event, the current activity goes to an exception state, for an “eventually” event the current activity continues on its course. Immediate events have the keyword “immediately” in their related reaction rule(s).

When events that are the start of Direct Flow are triggered, a token is sent over that flow to the connected activity. In case of an event with a PIC, that constraint becomes active when the event is triggered in case of “immediate” or after the activity is done in case of “eventually”. Direct Flow connected to “immediate” events changes the course of the current token, leading it to the activity connected to that flow, Direct Flow with “eventually” events splits the token, sending it along its original path as well as over the flow connected to the event, parallel to the normal flow.

When using process-wide events with direct flow, all activities should have an outgoing flow to that event, showing a possible path for the token to travel. This would immensely clutter the model, making it very hard to comprehend. But since a process-wide event is always connected to all activities (otherwise it would be activity-based) there is no need to actually draw this flow towards it. Process-wide events are therefore floating, implicitly linked to all activities. Tokens can therefore “jump” (immediate) or “copy” (eventually) towards such an event.

Activity <> Process	Immediate <> Eventually	Direct Flow <> PIC	How to model	Example
Activity	Immediate	Direct Flow	Event on activity border, Token only on connected outgoing flow	If documents that need to be checked are missing, the client is informed (figure 7.4)
Activity	Immediate	PIC	Event on activity border, activity goes to exception state	If a hotel is unavailable, the booking is stopped and “check flight-dates” must be executed again
Activity	Eventually	Direct Flow	Event on activity border, Token also on connected outgoing flow	After a timer event is triggered, the customer should be informed about the delay, but the process also continues
Activity	Eventually	PIC	Event on activity border, activity continues in current state	If required information is unavailable, the activity “accept customer” becomes impossible to execute
Process	Immediate	Direct Flow	Event floating in model, implicitly connected to all activities, Token “jumps” to event when triggered	If fraud is detected somewhere during the process, the current activity stops and the police is warned immediately.
Process	Immediate	PIC	Event floating in model, current activity goes to exception state	If fraud is detected, the activity “accept customer” immediately becomes impossible to execute
Process	Eventually	Direct Flow	Event floating in model, Token “copied” to event when triggered	If an overall time limit on the process is triggered, the customer is notified of the delay
Process	Eventually	PIC	Event floating in model, current activity continues in current state	If an appeal is filed, the claim should be examined again. (figure 7.6)

Table 5.3: Event modeling possibilities

5.5. Conclusion

In this chapter we looked at process modeling in the broadest sense. From defining what a process is to how you can look at it and how to model it. A modeling spectrum was described ranging from purely procedural till purely declarative models and some examples were placed throughout this spectrum. A new modeling language AREF was introduced and described. This language contains the possibility to model process transitions procedurally as well as declaratively, but some parts are required to be modeled declaratively (all decisions and calculations). It uses inspiration from BPMN

as well as ConDec to do so. AREF supports the use of all types of business rules (integrity constraints, reaction rules, derivations and authorizations), making it a total package for rule-based BPM.

6. Choosing the way to go

The previous chapter showed what the possibilities are for modeling a process: the spectrum from procedural to declarative and the options in between. Since AREF covers a big part of this spectrum, the choice of where to position a process has to be made. This chapter looks at the criteria for choosing the way to go for a specific process: when should you choose what place in the spectrum? What are the reasons for choosing that point?

In the AREF modeling language there is a choice between modeling in a completely declarative way, using only activities and rules, or a mixed approach using rules for decisions but modeling the rest of the process in a procedural way. This means that rules are always important in this approach, but the modeler has a choice in exactly how important they are. As discussed before, more rules instead of direct flow leads to more flexibility in the execution of the model, but makes understanding the design and its consequences harder to grasp. The upside is that this complexity in understanding is about the visualized, complete model of the process, not the involved verbalized rules or the execution of the actual work, making it less of an issue. This is because flow is better understood when visualized and rules are better understood when verbalized.

A general remark for all process modeling is to keep your audience in mind. If the model is only used to communicate the steps in the process, the choice between procedural or declarative depends on which one is best understood by the audience. If executable models are the goal (as they are here) other factors become important:

Which approach is chosen should primarily depend on the amount of *knowledge work* in the process. If a knowledge worker would know better than a system, declarative is the way to go. Actions can still be constrained, but with more freedom than a procedure. Other criteria for this choice are all connected to this concept of knowledge work: the complexity of the process, the unpredictability. The more complex a problem is, the harder it is to predict all possible paths and defining an exact procedure. If exceptions occur that have not been thought of in advance at design time, then the procedural model does not have the flexibility to deal with this. A procedure then becomes a straightjacket. Things are different, however, when knowledge work is codifiable. When something is called knowledge work because there is only one employee who knows how to do it, but this information can be automated, then it is not really knowledge work. The more complex, unpredictable and based on uncodifiable knowledge a process is, the more a declarative approach is the way to go. This is most obvious in a process that have different “paths” based on a knowledge-based decision early on in the process. The example used by Pesic (Figure 5.4: ConDec model for fractures treatment) is an example of such a process. In the initial examination, the doctor decides what the treatment plan will be (which path), but this could include some activities that have to be executed during the decision making process (X-ray, for instance).

Change intensity is another criterion to help make the modeling choice. Change intensity is about the frequency and amount of change in the process (model). This criterion is also influenced by which level in the model the change occurs on. If change intensity is high involving the rules inside an activity this has no impact on the modeling choice, because these rules are already flexible. If

change intensity is high on transitions between activities this has different consequences depending on whether the process is knowledge work: if the process is partially complex, unpredictable, knowledge work but not enough so to choose declarative, a high change intensity could tip the scale towards doing it declaratively. This is because with every change in a procedural model all possible exceptions have to be thought of and modeled, making the design effort bigger.

A third important thing to keep in mind is something that should not influence the decision, but could give problems in practice. This is the *modelers background and experience*. The problem with this is modeling by Business Analysts that have a business background as opposed to a technical background, or other non-technical modelers. The PICs can be quite hard to grasp for someone without a background in logical reasoning. Especially the response and precedence constraints can be confusing because they resemble direct flow. Even though this has nothing to do with how a process *should* be modeled, it will have effect on the success of a process model. Modelers will need training in order to work with the complexities of declarative modeling.

Table 6.1 sums up the criteria and recommendation for choosing between the procedural and declarative modeling options for executable models. Every situation has specific properties that might change the decision and the criteria are not black and white, but these recommendations can be used as a general guideline.

Knowledge work	Change intensity	Recommendation
Yes	High/Low	Declarative
Partially	High	Declarative
Partially	Low	Procedural
No	High/Low	Procedural

Table 6.1: Recommendations for modeling choices in AREF

7. Case study: processes in finance and insurance

This chapter describes a case in the world of banking and insurance. Three different processes are examined. Where possible, both procedural as well as declarative modeling in AREF and their pros and cons are shown. The chapter ends with some conclusions.

This case concerns processes in consumer finance and insurance for a well-known bank. In this time of financial crisis, these processes are under a lot of scrutiny. Supplying credit is one of the important processes. Assessing the risk loans is, next to rules and regulations, based on the experience of the loan officer, their gut-feeling so to say. Does this important and volatile process have to be built on strict procedures, or is it possible to place this authority into the hands of the knowledge worker? Because of this question, these processes are suited to be looked into in this case. In insurance, claims have to be examined very carefully in order to determine who gets what. This is also based on interpretation of rules and regulations by a knowledge worker and therefore interesting to examine here.

A number of services are provided by the bank in this case, including a finance and insurance center. The consumer finance and insurance center consists of a number of reception desks and private rooms in the front office, and a back office running supporting processes. They sell and handle three main types of products: personal loans, several types of mortgages and insurance policies. The front office handles sales and customer support, the back office handles all supporting processes for these products. Two of the consumer finance processes involved and one of the insurance processes are examined in this case:

- The loan sales process (primarily front office)
- Mortgage application assessment (back office)
- Insurance claim handling (back office)

Each process has its own activities, although they could be reused in other processes. For each of these processes the activities are summarized, then a direct flow and PIC model in AREF are shown (if both are possible) and the pros and cons of the choice examined.

7.1. The loan sales process

The main process for the customer finance center starts when a client walks through the door and ends with the signed up or rejected client leaving. Between the start and end of this process a number of activities can take place. These are:

Activities	Explanation
Receive client	A client visits the consumer finance center and reports to a reception desk
Identify client needs	The client's needs are identified by asking questions, after this a follow-up is chosen

Advise client	The client is advised on available products and procedures
Fill out application	The applicant is asked to provide the required data
Scan documents	In case the client has delivered documents on paper they have to be scanned into the system, because the back office checks them
Check documents	The documents are checked if sufficient for the application
Risk assessment (sub-process)	The application is assessed on the level of risk involved
Inform client	The client is informed about a deficiency in the documents
Signup client (sub-process)	The client is signed up for the loan. The actual loan process is started
Reject client (sub-process)	The client is rejected and this has to be registered in the system

Table 7.1: Activities in the loan sales process

The first three activities route a client to the product or procedure of his choice. The next activities are about that product, in this case a loan. In order to be approved for a loan two things have to be done. Required documents have to be delivered by the client and checked by the bank, and a risk assessment has to be performed on this application. This leads to signing up or rejecting the applicant.

7.1.1. The models

Whether the model is made with direct flow or PICs does not affect the other business rules involved, such as the state integrity constraints and resource rules, so they can be defined separately. This will follow after the differences between the two models.

First up is the PIC model. Because this model constrains where the model cannot go, choosing where the process does go is up to the person executing it. The first three activities are about routing the customer to where (s)he should go, leading to the model in

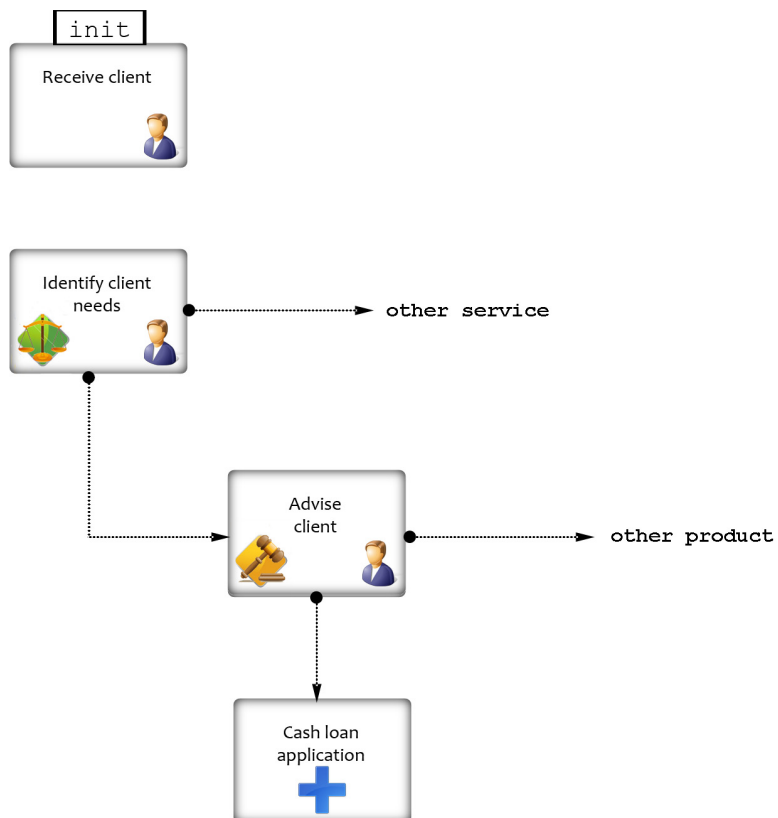


Figure 7.1: PIC model for routing the customer to a cash loan

Figure 7.1: *Receive client* is the initial activity, but depending on the client, all other activities are possible. If *Identify client needs* is chosen, the system makes a decision based on questions the client is asked. Depending on that decision the other PICs become active or not. If a client enters and tells the desk clerk (s)he wants a cash loan, this activity can immediately be started. What happens here is shown in Figure 7.2: Expanded PIC model for a cash loan. The *Cash loan application* sub-process is shown on blue.

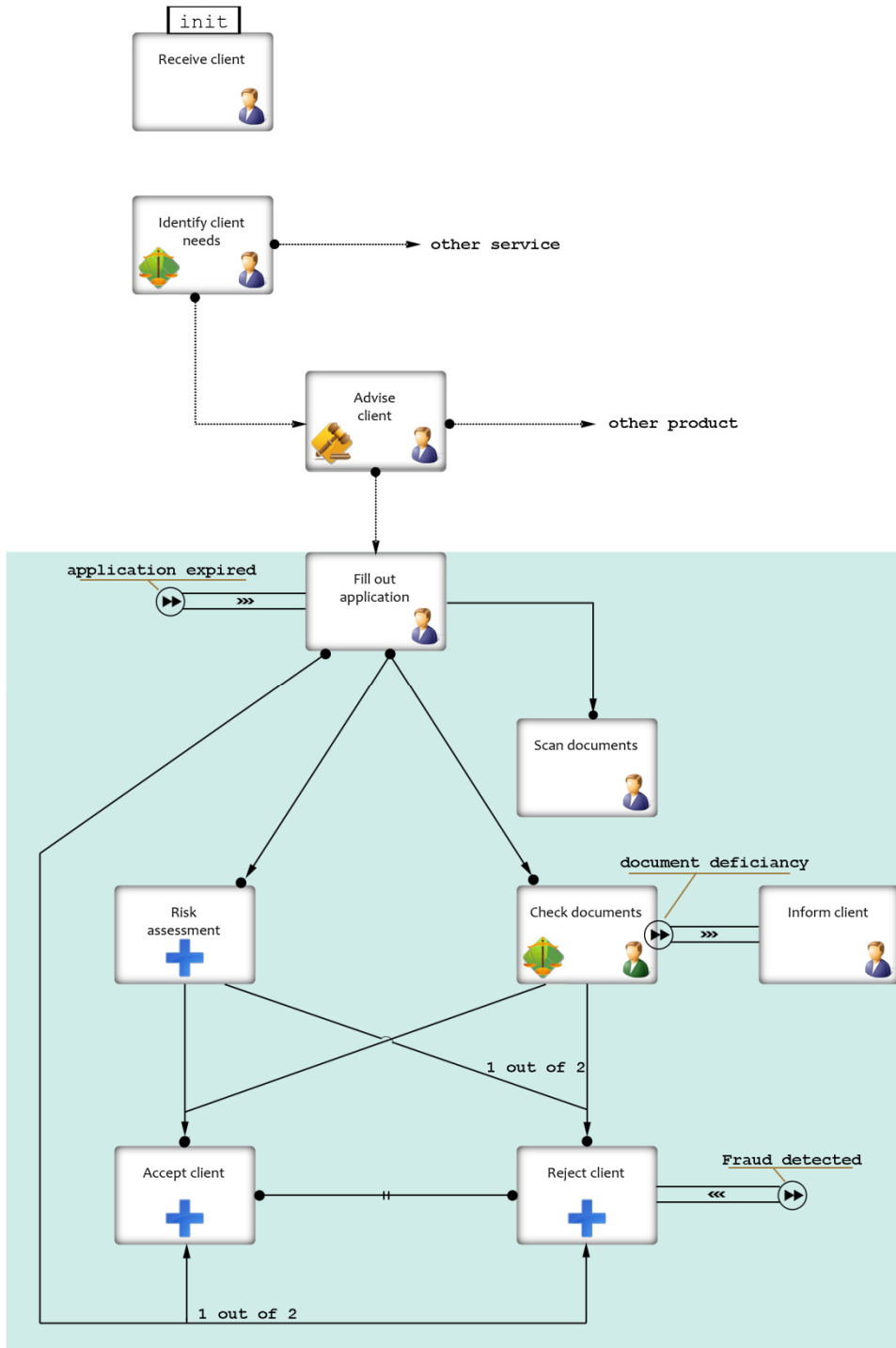


Figure 7.2: Expanded PIC model for a cash loan

After the application is filled out, *Check documents* and *Risk assessment* (both succession to filling out the application) define whether a client is accepted or rejected (which is mutual exclusive). One of these two has to follow *Fill out application* eventually (response). If the documents need to be scanned, this has to be done after filling out the application (precedence) in order to link the scanned files to a client. Before a client can be accepted, both *Risk assessment* and *Check documents* have to be completed (precedence), but a client can be rejected after only one of these activities is completed (precedence). There are two conditional constraints before *Accept client* and *Reject client*. If the risk and documents are approved, *Risk assessment* and *Check documents* has to be followed by *Accept client* (response), otherwise it has to be followed by *Reject client*

There are three events possible in the process. The first one is for fraud detection. When this occurs the client is immediately rejected (direct flow constraint). The precedence constraint on *Reject client* is then overridden. Second event is when it turns out there is something wrong or missing in the documents. When this happens the client is informed (direct flow constraint) and (s)he has time until the application expires to solve this issue. The rest of the process (for instance *Risk assessment*) can still continue. When this application expires it is cancelled and the process has to resume with filling out a new application (direct flow constraint). All activities currently running at that time will go into the cancelled state.

Now for the direct flow model. Routing the customer stays more or less the same. Only difference is that the routing activities cannot be skipped. For the *cash loan application* itself the simplest solution is defining an order in which the activities have to be done. This leads to the second model.

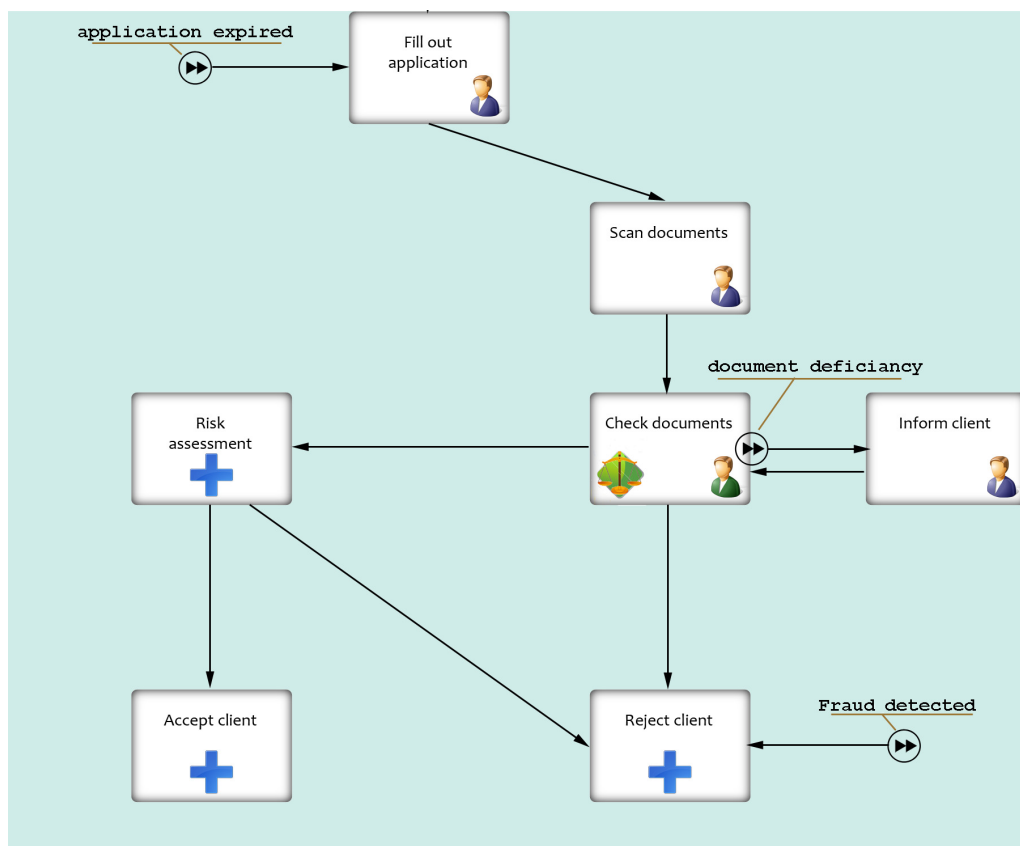


Figure 7.3: Cash loan application simple direct flow model

The downside of this model is that activities like *Scan documents* are often skipped and if the client does not have the documents available to be checked the process grinds to a halt, making it a bottleneck. The bank doesn't want this delay and wants to offer the service to check the involved risk (*risk assessment*) first if the documents are not available right away. To facilitate this an extra activity has to be added. This activity decides which activity has to be done next, so it connects to conditional flow. This activity is called *Case routing* and can be seen as a collection of yes/no gateways in one activity.

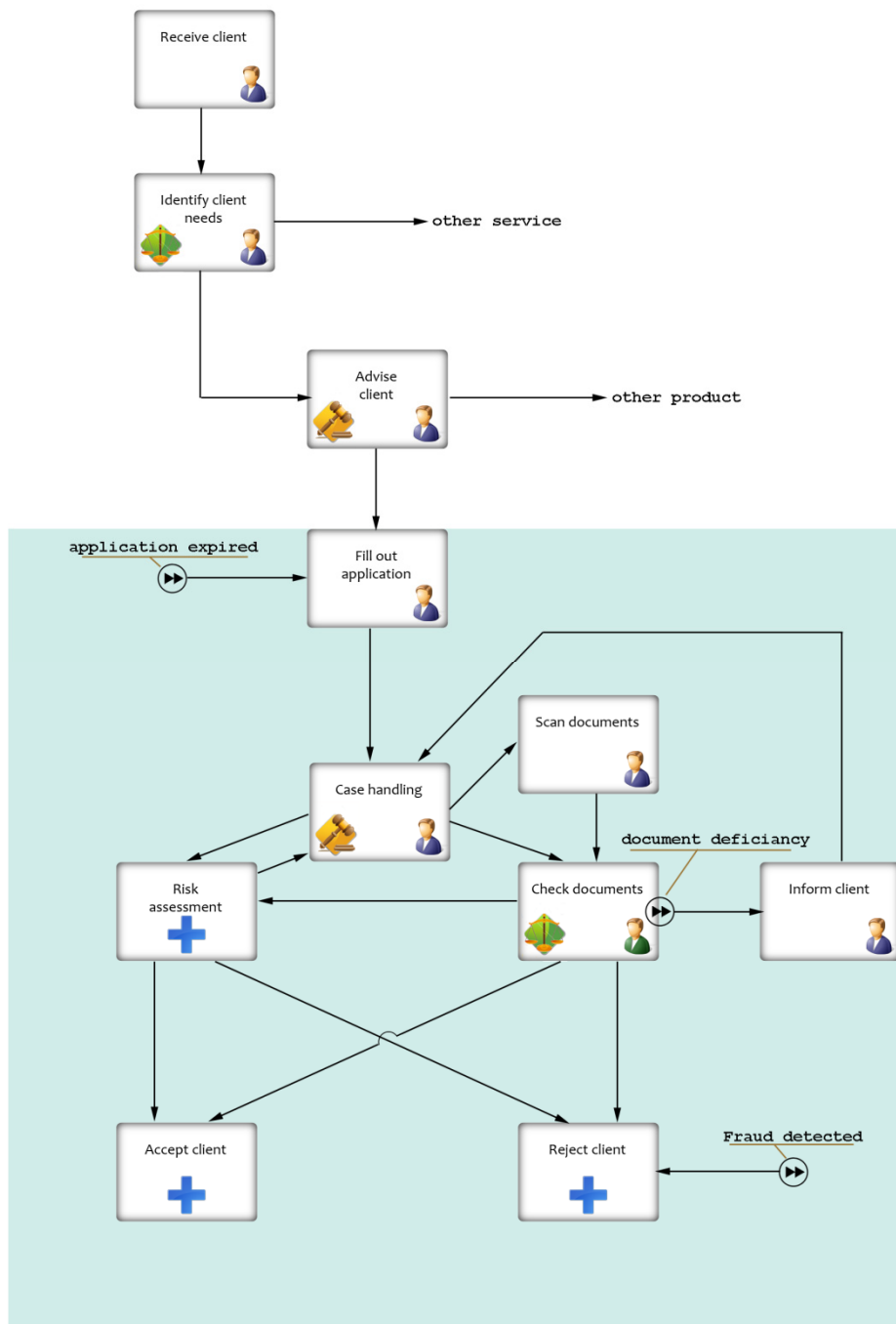


Figure 7.4: Direct flow model for a cash loan

This model gives the user the freedom to first check the documents or first do the risk assessment, avoiding getting stuck with a next required activity and following the intended process better than the simple direct flow model.

Receive client rules:

Process Integrity Constraints:

Start with *Receive client*.

Resource rules:

Receive client must be executed by Front Office clerk.

The other rules involved in this process are linked to the activities. These are not directly visible in the model, only when selecting a certain activity are they shown in a side screen²⁷. This screen shows all rules relating to that activity, including the PICs (if applicable). They are sorted by type: PIC, SIC, resource and reaction. Some activities have very few rules, like the *Receive client* activity (Table 7.2: Receive client rules), while other (mostly decision activities) have many rules. The number of rules and therefore strictness of the model is up to the modeler. An

Table 7.2: Receive client rules

example of an atomic activity with more rules is the *Fill out application* activity (Table 7.4: Fill out application rules) and the rules involved in the decision activity *Check documents* can be found in Table 7.3: Check document rules.

Fill out application rules:

Process Integrity Constraints:

If cash loan is chosen, *Advice client* must be followed by *Fill out application*.

Fill out application must be followed by *Check documents* **and** *Risk assessment*.

Fill out application must be followed by *Accept client* **or** *Reject client*.

Check documents cannot be executed before *Fill out application*.

Risk assessment cannot be executed before *Fill out application*.

Scan documents cannot be executed before *Fill out application*.

Resource rules:

Fill out application must be executed by Front Office clerk.

State Integrity Constraints:

Personal_ID, Name, Date_of_birth, Address, Nationality and Client_nr must be filled in.

The application must be printed and signed by the client.

Table 7.4: Fill out application rules

²⁷ It is important to show the rule list next to the model at the same time and not in completely different screens in order to keep the integrated feel of activities and all rules.

Check documents rules:**Process Integrity Constraints:**

Check documents must be followed by *Accept client* or *Reject client*.

Check documents cannot be executed before *Fill out application*.

Accept client cannot be executed before *Check documents* **and** *Risk assessment*.

If documents are approved **and** risk is approved, then *Check documents* must be followed by *Accept client*.

Reject client cannot be executed before *Check documents* **or** *Risk assessment*.

If documents are not approved **or** risk is not approved, then *Check documents* must be followed by *Reject client*.

Resource rules:

Check documents must be executed by Back Office clerk.

Check documents uses client documents.

State Integrity Constraints:

Documents are approved if

Photo identification (Passport, ID or Drivers License) is supplied and valid.

All mandatory fields in the application document are filled out and the document is signed by the client.

Credit check consent form is filled out and signed.

Recent bank statement is supplied.

Reaction rules:

If there is a document deficiency, then immediately activity state becomes paused and go to *Inform client*.

Table 7.5: Check document rules

Each process-wide event can also be selected to show its reaction rule(s). For the *application expired* event that rule would be: "Immediately go to *Fill out application* if the *application expiration date* expires."

7.1.2. Pros and cons

For the direct flow model, the pros are that it is easy to understand how the steps in the process follow each other and all flexibility is transparent because the only flexibility is in the rules of the decision activities. All possible paths are visible by following the direct flow. Cons are that all possibilities have to be modeled at design time making the model inflexible. For example, when *Risk assessment* is done first, the model has to go back to *case routing* because the user has to determine whether the supplied documents have to be scanned or not. This extra *case routing* activity is not a natural way to model this work. More natural would be to model it at the end of every activity, because this is when a user thinks about what to do next. (S)he knows if a client is standing there with paper documents or that they are supplied digitally, no extra activity is necessary for deciding if scanning is required. The strict direct flow model has no way of modeling this.

For the constraint model pros are opposite to the direct flow model. With constraints it is possible to model users choices in the process in a natural way. By giving the user a list of possible activities choosing to start *scan document* becomes a decision that does not need modeling, but is made by the user based on the knowledge (s)he implicitly has (for instance seeing a client holding the

documents). Another pro is the flexibility in the model. If a client walks in and the first thing (s)he says is: I want a cash loan, there is no need to do the *Identify client needs* and *Advice client* activities. The PIC model gives a user this freedom. Con is that the possible flow in the model is not instantly clear when looking at the model. It takes some learning to be able to understand the consequences of this type of modeling, and even then it is harder to foresee all consequences than with the direct flow model where you only have to follow the path. Some constraints involving arrows can easily be mistaken for flow by an inexperienced user.

7.1.3. The author's choice

Based on the criteria from the previous chapter the direct flow model from figure 7.4 is probably the best choice here. The process itself is relatively simple and does not involve a lot of knowledge intensive work, which requires flexibility. Even if the client knows what (s)he wants, the employee can easily skip the screens for the routing activities and move on with the *cash loan application* and facilitating the handling of that process with a separate *case routing* activity and screen is not a big problem. Because the decision activities are rule based, these offer the required flexibility. This case routing approach eliminates the work list and the user interface to work with that, because all screens for activities can be linked together in the way the model depicts. This case routing is not complex or knowledge intensive, but similar to complex gateways used in BPMN. Change intensity is probably low, so this does not increase the need for a declarative approach. The simple direct flow model from Figure 7.3 would not be the best choice here, because it is too rigid, not allowing for a quick execution of instances where the documents are not immediately present.

7.2. Mortgage application assessment

In the previous process, the *risk assessment* sub-process was not elaborated on further. This is a small sub-process which has one key activity: execute risk assessment. This activity is where the actual decision is made and this is a comparable decision to assessing a mortgage application, another type of loan. There are a lot of these decisions to be made in a financial organization and these can all be designed similarly. This assessment activity could be a service that is used in different processes, like selling a loan versus selling a mortgage, only with different rules. Defining aspect of such an activity is that a yes/no decision has to be made on the basis of a lot of data. Most of this decision can be captured in Business Rules, advising the person making the decision. Sometimes it can even be completely automated. So there is only one activity in this process leading to a yes or no answer: *Execute assessment*.



Figure 7.5: Execute assessment rule-supported decision activity

What direct flow, PICs or resource rules apply to this activity depends on the surrounding process. This is done in the same way as in the previous section and will be left out of the account here. What is interesting is how this decision is made and what rules apply.

If a decision needs to be completely automated, there is no room for flexibility in the decision, it is black or white. Rule-supported decisions allow for some grey areas where a person decides the outcome, based on his or her knowledge. So again, this is a question about flexibility (like with the choice between flow and process

constraints). Even though all decisions are done declaratively, a rule-based (automated) decision has no flexibility and all possible outcomes have to be thought of in advance. The bank prefers to keep this decision in the hands of a person, so a rule-supported decision is chosen.

To keep the example simple, the client knows the amount (s)he needs to buy a house and what type of mortgage (s)he wants, making this a yes/no decision. The rules for this decision mainly depend on the income of the applicant. If the type of mortgage is uncertain, more rules will apply. The rules are used to calculate a maximum mortgage sum for the applicant and the claim handler decides whether the application can be approved. In real life, there are more rules involved in this calculation, but these would make the example less simple without adding any real value to understanding how this decision works. The maximum mortgage sum depends on the *gross total income (gti)* of the applicant, its *annual growth*, the applicants *capital* and *credit rating*. The *gti* depends on whether the applicant is married, because then the spouse's income is added to the applicant's, leading to a higher *gti* and therefore a higher *mortgage sum*. There are also some rules regarding a *mortgage guarantee* about the *mortgage sum*, the *house* and the applicants *employment contract*.

Rules for mortgage applications

Mortgage sum must be under $(gti * 4,5)$

or $(gti * 4,5) + 20\%$ if *annual gti growth* is over 10%

or $(gti * 4,5) + 25\%$ if *capital* is over *gti*

or $(gti * 4,5) + 40\%$ if *capital* is over $gti * 2$

or $(gti * 4,5) - 10\%$ if *credit rating* is B

or $(gti * 4,5) - 30\%$ if *credit rating* is C

Gross total income is *gross income applicant* + *gross income spouse* if married

A *mortgage guarantee* can only be included if

mortgage sum is under € 265,000

and *house* is *primary residence*

and *house* needs less than 10% of its *price* in remodeling

and *employment contract* is *permanent*

Table 7.6: Mortgage application rules

According to the rules in Table 7.6: Mortgage application rules, an *unmarried* applicant that needs € 200,000 to buy a house as a *primary residence* that *does not need remodeling*, has a *gross total income* of € 38,000 a year in a *permanent* job and wants the *mortgage guarantee* will not get this mortgage unless (s)he has an *annual gti growth* over 10% or at least € 38,000 in *capital* supported by a *good credit rating*.

7.2.1. Pros and cons

Biggest con of this approach to making decisions is that there is no easily followed decision process beforehand. The example is kept fairly simple, but it might be less transparent in real life where the number of rules is much higher. The mentioned black box problem occurs here, where people do not trust the outcome of something they cannot see. This problem can be solved by having the system generate a decision trace. This trace shows how the decision was made and makes it transparent. Pros are clarity of the rules and data involved in this decision and possible automation of the process

or parts of it. Even when a person makes the actual decision, the outcome of the system and what rules this was based on can be saved with the decision trace. If rules change over time, saving old versions of the rules will give users the ability to recreate the circumstances that were involved at the time of a decision in the past.

7.2.2. The author's choice

In case of a yes/no decision like this one, AREF offers only one way of modeling: the decision activity. There is no choice between procedural or declarative here, these decisions are always declarative. The only decision here is: to make the decision rule-supported (a person still makes the actual decision, supported by the outcome of the rule engine) or to make it rule-based, completely automating it. In this case the knowledge of the bank employee is required to look at factors that are not captured in the rules.

7.3. Insurance claim handling

Handling an insurance claim is a job for the back office. There are three possible outcomes for a claim: it is paid in full, partially or the claim is denied. This looks a lot like the assessment in the previous section, but there is a small, but significant, difference: in the process of coming to a decision on this claim, there are a number of activities that can or must take place. These activities are:

Activities	Explanation
Examine claim	Examine the claim
Negotiate deal	Negotiate a deal between the parties to solve the claim
Consult expert	Consult an expert opinion, for instance a damage, medical or legal expert
Approve large payment	Large payments need to be approved by a senior manager
Handle fraud	In case of fraud, this sub-process is started
Deny claim	The claim is denied, sub-process that notifies the client, registers the denial, etc
Pay partial compensation	The claim is partially paid, sub-process that notifies the client, registers the payment, does the payment, etc
Pay full compensation	The claim is paid in full, sub-process that notifies the client, registers the payment, does the payment, etc

A number of these activities are required, some are optional and some cannot be done in the same process instance. When, for example, a client is involved in a car crash and the two parties blame each other, a car damage expert is probably needed to assess a claim and it might be possible to

negotiate a deal. This process could be somewhat different each time and depends on the knowledge of the handler. The sub-processes for handling fraud or paying the compensation are not elaborated on further. In this process it was chosen to handle fraud inside the process. Another option would have been to define a fraud-event on the edge of this entire process and go to an exception state when fraud was detected. In this case it is one out of four modeled outcomes of the process. The outcome of the process must be one out of four: the claim is denied, the claim is partially paid, the claim is fully paid or fraud is detected. In case the claim payment is above a certain amount (for instance € 10,000), it has to be approved by a senior manager. The process can be disrupted when an appeal is filed against a decision. In that case the claim is re-examined.

7.3.1. The models

Since knowledge work and flexibility is involved the PIC model is the most obvious one to start with.

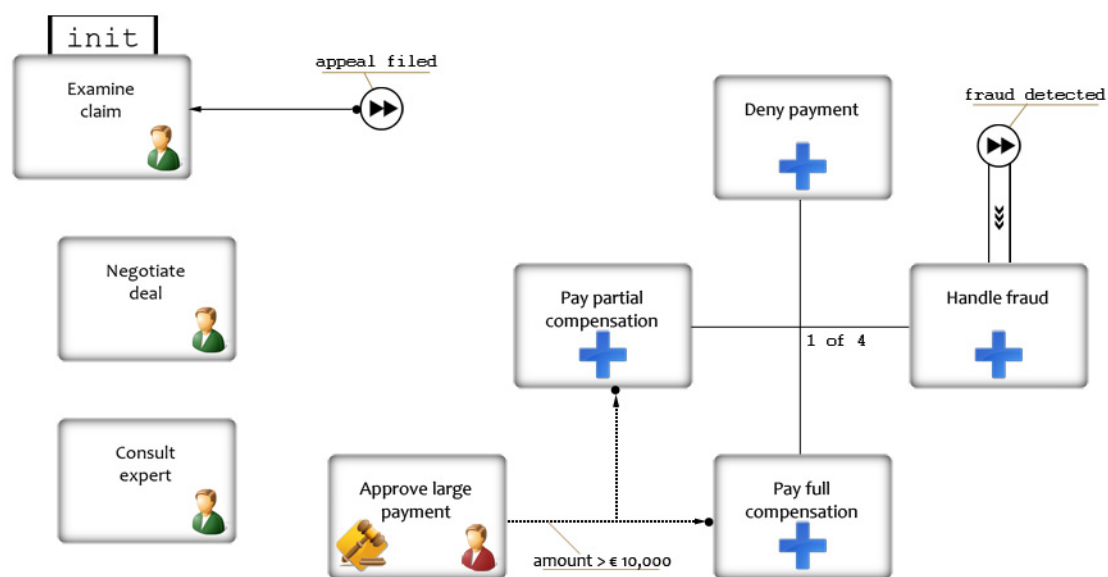


Figure 7.6: Insurance claim handling PIC model

All claim handling must start with an examination of the claim, so the *init* constraint is used. The *1 of 4* constraint is a combination of two types of constraints. One is that one of these four is required for the process to end, the other is that all four are mutually exclusive. Combined, this means that exactly one must be executed. The conditional constraint that *Pay partial compensation* and *Pay full compensation* have to be preceded by *Approve large payment* is only enforced with amounts over € 10,000. When an appeal is filed, *Examine claim* has to be done again, but this could be preceded by another activity if deemed appropriate. If fraud is detected, the process has to instantly go to the *Handle fraud* activity (direct flow constraint). There are no other constraints, so the process can involve activities like *Negotiate deal* and *Consult expert* if needed, but this is up to the handler of the claim, making this model very unrestricted.

A direct flow model for this process with the same flexibility is almost impossible to make. It would require a case routing activity like in the loan sales process, but this would not be for simple routing to one of three activities based on simple rules. There would be hardly any rules involved and the

case routing activity would be linked to almost every other activity in the process, like a spider web (Figure 7.7: Illegal insurance claim handling direct flow model).

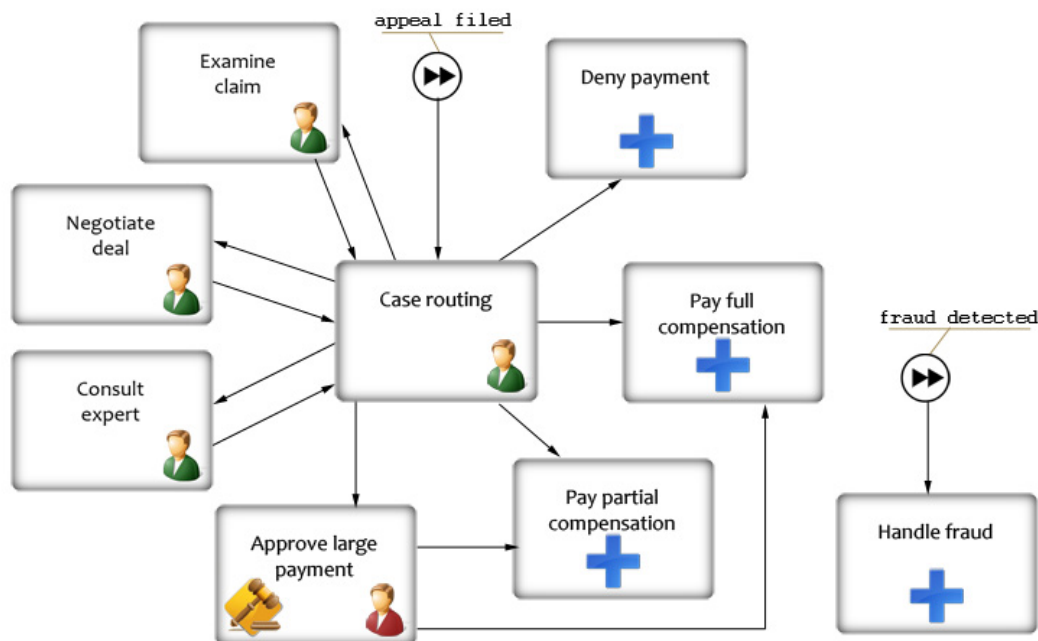


Figure 7.7: Illegal insurance claim handling direct flow model

Without rules doing the routing it is no longer a rule-based or –supported decision. In AREF, conditional flow can only connect to decision activities, making this an illegal model and therefore not allowed. The decision to start with the *Consult expert* activity will be based on the knowledge of the user. When this is captured in a rule, for instance *Consult expert must be executed if the user does not have the expertise*, this rule is not a natural Business Rule for this process. It is only needed because of the modeling approach. The routing rules from the loan sales process in section 7.1 are simple and the same for each user: documents are either available or they are not and this influences the process. In this insurance process the routing is the knowledge work, and can therefore not be captured in rules.

Another way of modeling this process with direct flow would be pouring the process into a strict procedure. Choices on the order of the activities would have to be made and flexibility is lost. Like in the loan sales process, this is not something the bank is aiming for.

7.3.2. Pros and cons

The PIC model gives a very clear overview of the few rules on control flow that are involved. The direct flow model requires over-specification of the model. Another con of the direct flow model is that handling fraud inside the process is now modeled completely separately, making it no use to model it here, inside the process. The more obvious choice would then be to add the fraud event to the edge of the entire claim handling process.

7.3.3. *The author's choice*

Since the process would benefit from a flexible approach to each claim, this is a typical example of a process where declarative modeling is the way to go. Designing this type of flexibility in a direct flow model is illegal in AREF (because it is not rule-based) and the other direct flow option leads to too strict a procedure, it would be very over-specified. One look at the PIC model shows a viewer that there are four possible outcomes and that the process is very flexible, so this is the authors choice.

7.4. Conclusion

This chapter described the modeling possibilities in AREF for three real-life processes in banking and the choices one has to make with that. Recommendations have been made on when to chose what modeling options, but this can be different for each situation, based on the involved circumstances.

In practice, most of the processes can be modeled in more than one way. The direct flow model with the simple case routing activity and the declarative model for the loan sales process are pretty similar in flexibility. Also the mortgage assessment and the claim handling are both mostly decisions, and the two different ways they can be implemented are close as well. All in all, AREF gives a lot of freedom to model a process the way a modeler wants to. An important difference with other languages however is the rule-based nature of all models, stressing their importance.

8. Conclusions and future work

This chapter examines whether all research questions as posed in chapter one were answered and what the answers are. Subsequently, future work in this field is discussed.

8.1. Questions and answers

1. How can Business Rules contribute to Business Process Management?

a. What is Business Process Management?

Business Process Management is two things: first, a management discipline, containing the activities: *“the discovery, design and deployment of business processes, but also the executive, administrative and supervisory control over them to ensure that they remain compliant with business objectives.”* (Smith & Fingar, 2003). Second, it is used to describe BPM-enabling technologies: *“methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information”* (van der Aalst, ter Hofstede, & Weske, 2003). These two types of BPM can be spread across the three organizational levels: strategic, tactical and operational, where different parts of a complete BPM approach are important. BPM is involved with different types of processes, that require different support. Important parts of BPM are process models and the process life-cycle.

b. What are the drivers and issues of Business Process Management?

The business drivers for BPM are making processes within and across an organization more efficient, more effective, more agile and more compliant. Issues are that processes are not agile because process models are interpreted into applications, and changing the application takes too long. Compliance is also partially missed, because transparency of rules in a process is very low. This makes it hard to make people accountable if they do not really understand what they are signing off on.

c. What is the Business Rules Approach?

The Business Rule Approach is *“a methodology—and possibly special technology—by which you capture, challenge, publish, automate, and change rules from a strategic business perspective”* (von Halle, 2001). It is a way of explicitly capturing the rules of the business in non-technical, structured natural language (and/or its formalized counterpart). There are four types of business rules (integrity constraints, derivation rules, reaction rules and authorizations) and they need structural assertions (a fact model) to be built on. If properly formalized, rules can be executed by a rule engine and there are some guiding principles like RuleSpeak and SBVR to help define business rules.

d. What are the drivers and issues of the Business Rule Approach?

Drivers for the BRA are also agility and compliance, but another important goal to reach with business rules is awareness of the rules and regulations in the organization. Mass differentiation and some knowledge retention are also mentioned as reasons to work with the BRA. Issues with

business rules are the black box problem, where users do not trust or are uncertain about the rule system and whether it works as intended. Another issue is the inability of some users to define correct rules and something to watch out for is that rules have to be structured in such a way that the transparency they offer is retained.

e. Can a rule-based BPM approach help support both the BRA's and BPM's drivers?

Yes. Chapter four showed the overlap and collaboration BPM and the BRA have. Together they support all drivers the combined field has. The agility problems in BPM, like disintermediation, are solved by empowering business users with understandable rules and the transparency rules provide helps with the other parts of agility: reusability, progressiveness and integrability. Prerequisite for the success of this combination is the use of executable process models. The BPM issues with compliance are also aided by the transparency of rules, making users truly accountable, and by the explicit enforcement of rules and regulations.

2. What modeling options are there with rule-based BPM and when should each be used?

a. What is the difference between process modeling in BPM and the BRA?

Process modeling can have different perspectives, making different parts of a process most important. In BPM, modeling is done procedurally, specifying all possible paths. In the BRA, modeling is done declaratively, specifying all applicable rules. In the procedural style, rules are implicit and in the declarative style, flow is implicit. A declarative style gives more freedom in execution, allowing for flexible workflow. Because of this flexibility, not all exceptions have to be modeled at design time, but can be handled during execution of the process. Both styles have their strengths and weaknesses, making them suited for different types of processes.

b. Can their meta-process models be combined?

The meta-model for BPMN, the modeling standard for BPM, showed five important concepts: activity, flow, gateway, event and swimlane. The declarative ConDec language showed activities and constraints as the most important parts. The AREF modeling language combines these two by also making activities the main concept, offering a choice between flow and process constraints (PIC), capturing gateways in rule-based decisions, modeling events and defining state constraints (SIC) and resource rules. This way AREF offers the modeler a choice to work completely declaratively or partly procedurally.

c. Which approach should be used when?

The more knowledge work is involved in a process, the more a declarative modeling style is appropriate, because it allows for flexibility, giving the knowledge worker a chance to rely on that knowledge instead of being chained down by the system into a rigid procedure. The process' change intensity, the amount of change the process endures, can be used to tip the scale in one of the directions, where more change favors a declarative style. The modeler background in or experience with logical reasoning could create a need for training in declarative modeling.

d. *How does it work in the real world? (case study)*

Chapter seven showed the application of AREF on three real-life processes that showed the different choices one may encounter. Each approach has its pros and cons and the author's choice was given. The case itself is the answer to this question.

8.2. Future work

The case study in chapter seven is a theoretical exercise. What the approach and meta-model now needs is a test in a real organization. This would provide the data to improve the choices possible in the model based on best practices and might lead to removing some of the choices because one option is apparently better than another.

The combination of BPM with the Service Oriented Architecture (SOA) has been researched quite extensively, but the triangle of BPM, BRA and SOA could bring together all strengths of the three approaches, maybe using rules as a basis for Service Level Agreements (SLAs). This field could use more research.

An issue briefly mentioned in chapter six is that flow is more easily understood when visualized and rules are more easily understood when verbalized. This issue needs more research to determine what the best method is for modeling a declarative model, if a proper visualization for rules is really useful. A related issue is modeling of processes in general. Are flows the most natural way of modeling a process, or just the most habitual way? These are more general cognitive issues in modeling that need more research.

Future research could also be done of the use of declarative process modeling in practice. There are some tool vendors that sell knowledge based systems, allowing for some kind of real-time dynamic flow, but it has not been researched if these really implement declarative process modeling and if so, what principles they base this on.

It could also be interesting to examine the possibilities of using another language than LTL for defining the process constraints, which might lead to the discovery of other useful patterns.

9. Bibliography

Bandara, W., Indulka, M., Sadiq, S., Chong, S., Rosemann, M., & Green, P. (2007). *Major Issues in Business Process Management: An Expert Perspective*. The University of Queensland. School of Information Technology and Electrical Engineering.

Burlton, R. (2001). *Business Process Management: Profiting from Process*. Indianapolis, IN: Sams Publishing.

Chisholm, M. (2003). *How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering*. Morgan Kaufmann.

Davenport, T., & Short, J. (1990). The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, Volume 31 (Issue 4), 11-27.

Dimitoglou, G. (2004). *Business Rule Computation in Distributed Organizations (dissertation defense slides)*. Washington: The George Washington University.

Faget, J., Marin, M., Mégard, P., Owens, V., & Tarin, L. (2003). Business Processes and Business Rules: Business Agility Becomes Real. In *Workflow Handbook 2003* (pp. 77-92). Future Strategies Inc.

Gable, J. (2002). Enterprise Application Integration. *Information Management Journal*, Vol. 36 (No. 2).

Georgakopoulos, D., & Tsalgatidou, A. (1998). Technology and Tools for Comprehensive Business Process Lifecycle Management. *Workshop management systems and interoperability*. vol. 164, pp. 356-395. Istanbul: NATO advanced study institute on workflow management systems.

Goedertier, S., & Vanthienen, J. (2007). Declarative Process Modeling with Business Vocabulary and Business Rules. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops* (Vol. LNCS 4805, pp. 603-612). Berlin / Heidelberg: Springer.

Goedertier, S., Haesen, R., & Vanthienen, J. (2007). *EM-BRA2CE v0.1: A vocabulary and execution model for declarative business process modeling*. Leuven: FETEW Research Report .

Hammer, M. (1990, July-August). Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, pp. 104-112.

Hay, D., & Healy, K. A. (2000). *Defining Business Rules - What are they really*. Final Report, the Business Rules Group.

Lau, K.-K., & Vanden Bossche, M. (2002). Logic Programming for Software Engineering: A Second Chance. In *Logic Programming* (pp. 263-281). Berlin: Springer.

Loggen, R. (2008). Human Centric Processes – the next challenge for BPM. In *Trends in BPM* (pp. 54-67). Capgemini.

Pesic, M. (2008). *Dissertation: Constraint-Based Workflow Management Systems: Shifting Control to Users*. Technical University Eindhoven. Eindhoven: University Press Facilities.

Pesic, M., & van der Aalst, W. (2006). A Declarative Approach for Flexible Business Processes Management. *Business Process Management Workshops* , 169--180.

Pesic, M., Schonenberg, M., Sidorova, N., & van der Aalst, W. (2007). Constraint-Based Workflow Models: Change Made Easy. *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS* , 77--94.

Reid, R. D., & Sanders, N. R. (2004). Total Quality Management. In *Operations management* (2nd Edition ed., pp. 136-170). John Wiley & Sons.

Ross, R. G. (2003). *Principles of the Business Rule Approach*. Addison-Wesley.

Sadiq, S., Sadiq, W., & Orłowska, M. (2001). Pockets of Flexibility in Workflow Specification . In *Conceptual Modeling — ER 2001* (Vol. LNCS 2224, pp. 513-526). Berlin / Heidelberg: Springer .

Smith, H., & Fingar, P. (2003). *Business Process Management: the third wave*. Tampa, Florida: Meghan-Kiffer Press.

Taveter, K., & Wagner, G. (2001). Agent-oriented Enterprise Modeling Based On Business Rules. In *Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001)* (pp. 527-540). Springer-Verlag.

van der Aalst, W. M., ter Hofstede, A. H., & Weske, M. (2003). Business Process Management: A Survey. *Proceedings of the 1st International Conference on Business Process Management. Volume 2678 of LNCS*, pp. 1-12. Springer-Verlag.

von Ammon, R., Emmersberger, C., Springer, F., & Wolff, C. (2008). Event-Driven Business Process Management and its Practical Application Taking the Example of DHL. *Proceedings of the 1st iCEP08 Workshop on Complex Event Processing for the Future Internet*. Vienna: CEUR-WS.

von Halle, B. (2001). *Business Rules Applied*. New York: John Wiley & Sons .

White, S. A. (2004). *Introduction to BPMN*. Object Management Group.

Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., & Russell, N. (2006). On the Suitability of BPMN for Business Process Modeling. In *Business Process Management* (Vol. LNCS 4102, pp. 161–176). Berlin / Heidelberg: Springer-Verlag.

Zairi, M. (1997). Business Process Management: a boundaryless approach to modern competitiveness. *Business Process Management Journal* , Vol. 3 (Issue 1), 64 - 80.

10. List of abbreviations

AREF – Activities Rules Events and Flow

BAM – Business Activity Monitoring

BPD – Business Process Diagram

BPEL4WS/BPEL – Business Process Execution Language (for Web Services)

BPI – Business Process Improvement

BPLM – Business Process Life-cycle Management

BPM – Business Process Management

BPMN – Business Process Modeling Notation

BPMS – Business Process Management System/Suite

BPR – Business Process Re-engineering

BRA – Business Rule Approach

BRM – Business Rule Management

BRMS – Business Rule Management System/Suite

CEP – Complex Event Processing

DBMS – Database Management System

EAI – Enterprise Application Integration

ERP – Enterprise Resource Planning

HIM – Human Interaction Management

LTL – Linear Temporal Logic

OMG – Object Management Group

PIC – Process Integrity Constraint

SIC – State Integrity Constraint

SOA – Service Oriented Architecture

STP – Straight Through Processing

WfM – Workflow Management