

Insert here your thesis' task.

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF SOFTWARE ENGINEERING



Master's thesis

Leveraging Toolkits in BPM Application Development

Bc. Jaroslav Fibichr

Supervisor: Ing. Robert Pergl, Ph.D.

10th May 2013

Acknowledgements

At this place, I would like to express thanks to all who helped me to finish this thesis.

Firstly, I want to thank my supervisor, Ing. Robert Pergl, Ph.D. for his guidance and providing ideas for the thesis topic. Secondly, I would like to thank my dear friend and colleague Bc. Miroslav Dzurenko, for bringing me to the Business Process Management, for the big help with all the aspects of the BPM applications development, and also for his friendly support. Another important person, whom I want to express my thanks, is my IBM mentor and the best superior a person can get, Ing. Radek Šulc, for providing me the inspiration for the thesis and excellent professional leadership. Also, I would to thank all the members of the BPM project team at the Faculty of Information Technology (especially its leader Ing. Daniel Matocha MSc.), and also members of the Center of Knowledge Management at CTU for the cooperation and sharing the valuable knowledge during the BPM project implementations. I would also like to thank my brilliant proof-reader, Mgr. Kateřina Pavésková.

My greatest thanks goes to my parents because without all their support I would never have the opportunity to get so far in my studies. And for all her support, care and love, I thank my girlfriend Alenka Rozsypalová.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60(1) of the Act.

In Prague on 10th May 2013

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2013 Jaroslav Fibichr. All rights reserved.

This thesis is a school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Fibichr, Jaroslav. *Leveraging Toolkits in BPM Application Development*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2013.

Abstract

The development of BPMS products has recently undergone rapid development. One of the key factors was the launching of the new version of the BPMN 2.0 specification, which enabled the development of process models directly transferable into an executable form. This new approach changed the way of development of processes in BPMS. This thesis deals with the possibilities of leveraging the reusability in these new BPM systems. The author compared different approaches in some BPMSs, focusing on IBM BPM platform. On the basis of his practical experience from a case study, he managed to assess the degree of reusability of the evaluated components.

Keywords BPM, business process management, components, reusability

Abstrakt

Rozvoj BPMS produktů prodělal v posledních letech bouřlivý rozvoj. Jeden ze zásadních faktorů bylo uvedení nové verze specifikace BPMN 2.0, která umožnila vývoj procesních modelů přímo převoditelných do spustitelné

formy. Tento nový přístup změnil i způsob vývoje procesů v BPMS. Tato práce pojednává o možnostech využití znovupoužitelnosti v těchto nových BPM systémech. Autor porovnal různé přístupy v některých BPMS, se zaměřením na platformu IBM BPM. Na základě praktických zkušeností z případové studie vyhodnotil míru dosažené znovupoužitelnosti zkoumaných komponent.

Klíčová slova BPM, procesní řízení, komponenty, znovupoužitelnost

Contents

Introduction	1
Goal of the Thesis	2
Structure and Methodology	2
 I Theoretical Part	 5
 1 Introduction to BPM	 7
1.1 Definition of Business Process Management	7
1.2 Brief History of Business Process Management	9
1.3 Characteristics of BPM	15
1.4 Summary	35
 2 Reusability in BPM	 37
2.1 Introduction to Reuse Principles and Approaches	37
2.2 Asset-Based Development	40
2.3 Measurements and Metrics	47
2.4 Summary	52
 3 Assessment of Reuse in Selected BPM Suites	 53
3.1 Introduction	53
3.2 IBM Business Process Manager	54
3.3 Comparison of Reuse Approach in Other BPM Solutions . .	72
3.4 Summary and Comparison	75

II Practical Part	77
4 Case Study	79
4.1 BPM Programme at FIT	79
4.2 Approach to Reusability	81
4.3 Implementation of the Selected Reusable Assets	83
4.4 Experienced Issues	99
4.5 Summary	100
5 Evaluation and Recommendations	103
5.1 Measuring Procedure	103
5.2 Results and Discussion	105
5.3 Recommendations	111
5.4 Summary	113
Conclusion	115
Bibliography	117
A Acronyms	123
B Development Cost Estimations	125
C Contents of enclosed CD	129
Contents	130

List of Figures

1.1	Business Process Management Hype Cycle [38]	10
1.2	Early BPMS Evolution [40]	12
1.3	Emerging Business Process Management Suites Market	13
1.4	Gartner: Hype Cycle for Business Process Management, 2012 [19]	14
1.5	BPM Life cycle by Smith and Fingar [62]	16
1.6	BPM life cycle by Koster [44]	17
1.7	Process Map [27]	18
1.8	Business process modelled in BizAgi Modeler [10]	21
1.9	User interface design	24
1.10	Example of user task list in portal (Activiti)	25
1.11	Business Activity Monitoring dashboard (IBM BPM)	26
1.12	SOA Elements	29
1.13	Symbiosis of BPM and SOA	30
1.14	BPMN forms the interface between business and IT.	32
1.15	Core BPMN elements (in BPMN v1.2) [13]	34
2.1	Approaches to software reuse in time (based on [14])	38
2.2	An asset as a collection of artifacts [52]	40
2.3	Asset life cycle and governance	41
2.4	Asset use cases (based on [18])	42
2.5	Search for assets using advanced search fields	44
2.6	RAS Metadata Structure	46
2.7	Cumulative costs comparison	50
3.1	Evolution of IBM Business Process Manager [43]	54
3.2	Position of IBM Business Process Manager in Forrester Wave - BPM Suites, Q1 2013 [57]	55
3.3	IBM Business Process Manager - Architecture Overview	56

3.4	IBM Process Center shared repository	58
3.5	IBM Process Center main view	58
3.6	IBM Process Designer	59
3.7	IBM Process Designer artifacts	62
3.8	Definition of Person Business Object	63
3.9	IBM Process Designer - Adding a dependency	66
3.10	Toolkit dependencies	67
3.11	Toolkit sharing among dispersed Process Centers	67
3.12	Process Application artifact metadata	68
3.13	Scaling from BPM project to program	71
3.14	Bizagi Widget Store [10]	73
3.15	Activiti components [56]	74
4.1	BPM project implementation status	80
4.2	Object-relational mapper	86
4.3	Person Business Object definition in IBM Process Designer . .	87
4.4	Person - UML Class diagram	88
4.5	Person - database schema model	88
4.6	Basic DB data retrieval and mapping	89
4.7	ORM Toolkit architecture	90
4.8	SQLBusinessQuery object structure	91
4.9	Person Object - Inheritance	94
4.10	Person Business Obejct structure	95
4.11	Coach with Student, Employee and ExternalPerson bound to the same Coach View	95
4.12	Person Coach View Configuration Options	96
4.13	Thesis Topic Approval process - each of the human tasks require tracking of changes	98
4.14	Changelog - Dialog Box for viewing the tracked changes	99
5.1	Reuse efficiency evaluation process	104
5.2	Impact of asset changes on reuse efficiency	110

List of Tables

1.1	Key benefits of BPM, their outcomes and typical metrics (based on [16])	15
1.2	Possible negative consequences of adopting BPM and SOA apart (based on [37])	31
2.1	Facets of reuse by Prieto-Díaz [55]	39
3.1	Mapping asset use cases to IBM BPM components and features	69
4.1	Existing Toolkits in CTU Process Center, *) denotes mandatory Toolkit	83
4.2	Assets chosen for evaluation	84
4.3	Comparison of ORM approaches	93
5.1	Usage count of reusable assets	106
5.2	Derivation of mappings/queries ratio (k)	107
5.3	Reuse efficiency calculation for evaluated assets	109
5.4	Comparison of results with findings of Jacobson et al. [36] . . .	111
B.1	Evaluation of ORM - Approach I (non-reusable development)	125
B.2	Evaluation of ORM - Approach II (development of reusable asset)	126
B.3	Evaluation of ORM - Approach II (application of reusable asset): (IIa) shows evaluation of mapping definition, (IIb) shows evaluation of query calls	126
B.4	Estimation of Person CV - non-reusable development	126
B.5	Estimation of Person CV - application of asset	127
B.6	Estimation of Person CV - development of reusable asset	127
B.7	Estimation of Change Log - non-reusable development; c denotes number of activities in BPD with tracked changes	127

LIST OF TABLES

B.8	Estimation of Change Log - application of asset; c denotes number of activities in BPD with tracked changes	128
B.9	Estimation of Change Log - development of reusable asset . . .	128

Introduction

Many organizations have been continuously facing challenges regarding increasing productivity, efficiency, and flexibility. This difficult task may be fulfilled by introducing changes in their managerial approach. One of the most successful ways to achieve improvements is adopting Business Process Management (BPM).

Within the last few years, some organizations have started to utilize specialized software tools designed to support BPM. As the popularity of BPM grew, the number of vendors that produced Business Process Management Suites (BPMS) rapidly increased. However, the first BPM suites were primarily designed for IT and business users only defined the requirements for the process. That negatively affected the agility of the BPM project implementations.

In 2011, a significant change in the area was caused by the introduction of the latest version of the Business Process Model and Notation (BPMN) standard. In this version (2.0), for the first time it provided a specification that enabled to execute the process models directly, without additional translations into specialized execution language. That resulted in the using of a shared process model representation which facilitated cooperation between business and IT during the BPM project implementations.

The new way of implementing the executable processes made some BPMS vendors to come up with altered development tools. They are no longer based on traditional programming like their predecessors. Instead, the processes are developed through graphic-based integrated environments that make it possible even for an individual without any programming skills, such as business analysts, to design the process. Despite the indisputable advantages of this new approach, it neglects one important aspect of software engineering, which is reuse.

Recently, several vendors have invested into their BPMS products to provide capabilities enabling reuse in BPM. Nevertheless, as these features are relatively new, they have not yet been proven to provide increased efficiency of business process development.

Goal of the Thesis

The aim of the thesis is to examine reuse capabilities of modern business process management suites within a real BPM project. To achieve this objective, I defined the following partial goals:

1. Provide theoretical basis for Business Process Management and software reuse in its context.
2. Review selected BPMS products while focusing on reuse capabilities and detailed analysis of IBM Business Process Manager.
3. Demonstrate how reuse can be leveraged in BPM project implementation practice. Propose a metric and evaluate the achieved degree of reusability.

My personal motivation for choosing this topic for my master's thesis is a challenge that I have been dealing with in every day BPM implementation practice. I often asked myself how much effort I should invest in doing work better in the first place in order to save some time later. Via this thesis, I would like to verify whether doing so ever pays off.

Structure and Methodology

This thesis has been divided into two parts, theoretical and practical.

The theoretical part focuses on reviewing literature of various sources. With regard to the topic, I searched not only scientific and academic environment, but also sources from the commercial area that also provided some valuable insights. In addition, I have used my own experience gained from the BPM projects I took part in during my professional career.

The first chapter is devoted to Business Process Management. Basic terms are defined, and a brief history overview provides the foundations for understanding the current situation in the BPM area. The BPM life cycle and each of its phases is described to characterize its specific principles and requirements for associated tools. **The second chapter** introduces software reuse in the context of BPM. It begins with a definition of reuse,

motivation and taxonomy of different reuse perspectives. The methodology basis for practising reuse is described and a metric for evaluation reuse efficiency is derived. The goal of **the third chapter** is to compare different approaches to reuse in the existing BPM suites using the findings from the previous chapters. Three different BPMS products are reviewed while focusing on IBM Business Process Manager and development with reuse.

The theoretical part of the thesis serves as the basis for the practical part that contains two chapters. In **the fourth chapter**, a case study is conducted using the knowledge from the theoretical part. The setting for the case study is presented and a general approach to the reuse throughout the BPM projects is described. For the evaluation purposes, several cases of reusable assets are examined in detail, including analysis, design and implementation. Two different approaches to the implementation were chosen; once as a reusable asset and the second one was development in a non-reusable way. In **the fifth chapter**, these two approaches to asset development were compared and evaluated using the metrics proposed in Chapter 2.

Finally, **the Conclusion** summarises the contributions and achieved results of the thesis. Appendix A provides a list of the used acronyms and Appendix B contains the tables with the detailed development cost estimations.

Part I

Theoretical Part

Introduction to BPM

This chapter serves as an introduction to the field of business processes management (BPM). Basic terms such as business process or business process management are defined together with a short overview of the history and current situation of the area. To describe the characteristics of BPM and related issues, I go through the phases of its life cycle and provide an overview of the most important BPM standard.

1.1 Definition of Business Process Management

In order to understand business process management (BPM), it is necessary to understand what a business process is. A common problem in BPM is an absence of universal terminology. Terms are often used loosely and their meanings are often interpreted differently. Business process is one of those terms. Lindsay in [49] examined the history and followed the meaning of the term business process over the time.

One of the possible definitions was proposed by Davenport in [17]: *A process is a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action.*

In addition to this definition, some authors put the main principle into the relation with the outcome for the customer, as did Hammer and Chardy in [26]: *"A business process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer."* Thus we can say that a business process never exists without a purpose and

1. INTRODUCTION TO BPM

it should always bring some value to whom it is performed. In this respect, we can even distinguish *process* from *business process*, as did Voříšek in [66].

Now, we can move on the definition of business process management. As will be described in detail in the following sections, BPM has gone through dramatic development over the recent decades. Therefore, there exists almost as many definitions as there are BPM practitioners. One of the definitions covers the meaning in its full extent though:

Business Process Management (BPM¹) is *"a disciplined approach to identify, design, execute, document, measure, monitor, and control both automated and non-automated business processes to achieve consistent, targeted results aligned with an organization's strategic goals. BPM involves the deliberate, collaborative and increasingly technology-aided definition, improvement, innovation, and management of end-to-end business processes that drive business results, create value, and enable an organization to meet its business objectives with more agility. BPM enables an enterprise to align its business processes to its business strategy, leading to effective overall company performance through improvements of specific work activities either within a specific department, across the enterprise, or between organizations."* [7]

In short, the BPM is defined as a set of principles, methods and tools used to identify, design, execute, monitor and control business processes.

Although definitions of BPM differ, we can conclude that none of them mentions that BPM is not a technical method, but a management discipline. This is an important fact because business process management principles may be applied even without using any technologies and still bring the organization measurable added value [41]. This can be concluded simply because such initiatives had been put in practice long before the BPM technologies emerged.

BPM might be confused with other acronyms, such as Business Process Modelling or Business Performance Management. In this thesis, BPM is strictly used for Business Process Management.

Business Process Management System/Suite (BPMS) refers to a set of (software) tools that support the continuous process improvement efforts in the organization across all phases of the process life cycle.

¹BPM might be confused with other acronyms, such as *Business Process Modelling* or *Business Performance Management*. In this thesis, BPM is always used for Business Process Management.

1.2 Brief History of Business Process Management

Although the term Business Process Management is understood in a context of today's world, its core idea has been a matter of interest for a long time.

One of the first innovators in the field was Adam Smith (1723-1790) who described in his book "An Inquiry into the Nature and Causes of the Wealth of Nations (1776)" a production process of a pin in his factory. He proposed that it would be more efficient if the production process was divided into a set of simple tasks which would be performed by specialized workers, instead of having each one of them performing all the required activities to produce a pin. Smith came up with the conclusion that this kind of *division of labor* is the key for increased productivity. [63]

Another step in the evolution of BPM was made by Frederick Winslow Taylor (1856-1915). In his *scientific management* (also called *Taylorism*), he introduced a new approach to applying science to the process engineering and also management. That included (among others) analysis, logic, standardization of best practices, and work ethic.

Taylor's efforts inspired one of the most influential manufacturer of the beginning of the 20th century, Henry Ford. His mass production factory represented a revolution that significantly improved productivity by organizing manufacturing processes differently. By using new technologies, introducing the conveyor or systematic application of standardization of methods and tools, Ford achieved lowering costs alongside with an increase in production. [63]

As Gillot stated in [25], since this era, the main principles remained the same, only the tools and the methods have been improving.

After the Second World War, new attempts to improve process management emerged. In 1951, *Total Quality Management* concept was developed. It was focused on quality control of what was produced and the optimization of associated processes. It was followed in 1980s by a *Six Sigma* method that was created by Motorola to enable an improvement of the quality level evaluated based on statistical calculations.

The considerable advance of information technologies at the beginning of the 1990s influenced the foundation of the Business Process Reengineering (BPR) initiative. It all started when Hammer and Chardy wrote a book called "Re-engineering the corporation" [26]. The concept was based on a belief that a radical business process change should generate dramatic improvements in critical performance measures. Hammer claimed that it cannot be achieved by a simple automation of the current state of pro-

1. INTRODUCTION TO BPM

cesses but they need to be redesigned completely before that. However, the method got a bad reputation for being too invasive and resulted in many failed BPR projects. These often included an implementation of the Enterprise Resource Planning (ERP) systems that represented automated solutions incorporating optimized processes. However, introducing the ERP systems caused casting the process in the software thus make it rigid and impossible to modify.

In 2002, Smith and Fingar in "Business Process Management: The Third Wave" ([62]) explained why a third generation of business process management was born. The first wave was represented by Taylorism which utilized labor division. The second wave came with reengineering as one-time activity with costly ERP implementations. In the third wave, enabling changes became the primary goal. Although BPR and BPM share the same goal of analysing and redesigning business processes, the main difference was aptly summed up by Ko et al. in [42]: *Business Process Re-engineering calls for a radical obliteration of existing business processes, while its descendant Business Process Management is more practical, iterative and incremental in fine-tuning business processes*. The BPM hype cycle in Figure 1.1 shows a summarized view of how the process cycle has progressed over the last three decades. [38]

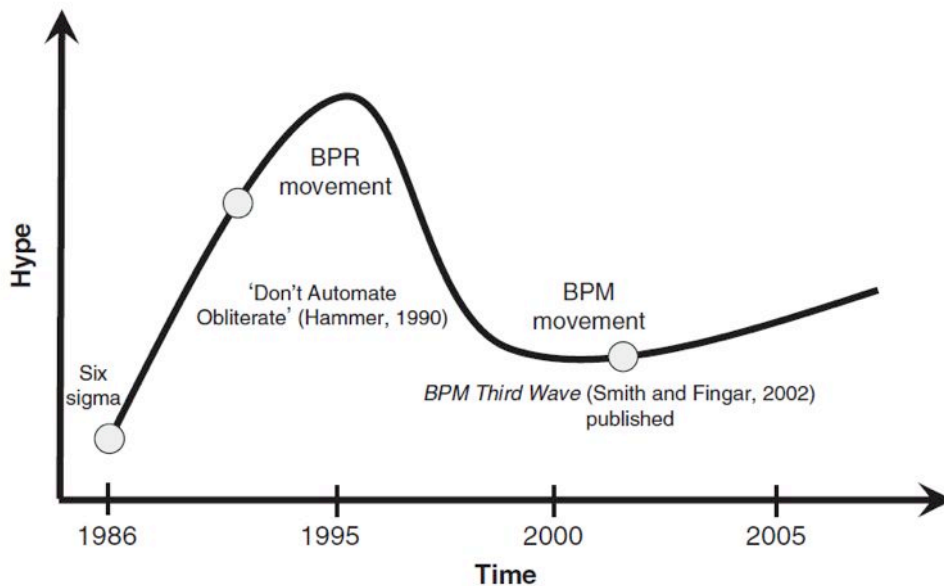


Figure 1.1: Business Process Management Hype Cycle [38]

During the past decade, the mainstream market of divergent BPM tools

has been shaped. Kemsley in [40] summarized the trends until 2006. He pointed out that two major predecessors of the BPM suites appeared:

1. Processes that require human involvement, therefore are executed within the workflow systems focused on capabilities like process monitoring, reporting, analytics and governance. However, none of these functions were generic, they needed to be custom-made. By developing new features, these systems eventually became "Pure-play" BPM suites.
2. Processes that can be handled automatically. Such systems were called *Enterprise Application Integration (EAI)* and they were used to automate the near-real-time exchange of data between systems. Since this does not accommodate any kind of human interaction it is often called *Straight-through processing*. The main focus was on fast and reliable messaging. Enterprise Application Integration systems finally became *integration-based* BPM suites.

Both approaches shared the same goal which was to reduce the flow time and cutting operational costs. Eventually, due to some mutual requirements for these systems, they blended into what we today call BPM suites. In such systems, benefits of both of their predecessors are combined into one integrated package. The evolution shifts are shown in Figure 1.2. Nonetheless, the distinction between the two approaches still may be seen among the currently offered BPM tools.

1.2.1 Current Situation

In 2013, after struggling to find its actual position within the organization, BPM became the primary platform to facilitate changes which very often have unknown and unforeseen consequences. Although the BPM suite space has gone through multiple waves of consolidation, there are still tens of vendors operating on the BPM suites market. All these competitors strive to bring the added value to the customer in their own specific way.

Prior to its latest market research report [57], Forrester¹ used to divide the broader BPMS market into 3 groups (see 1.3):

1. **BPM suites** represents former "human-centric" BPM suites

¹Forrester Research is an independent technology and market research company that provides advice on existing and potential impact of technology, to its clients and the public. <http://www.forrester.com/>

1. INTRODUCTION TO BPM

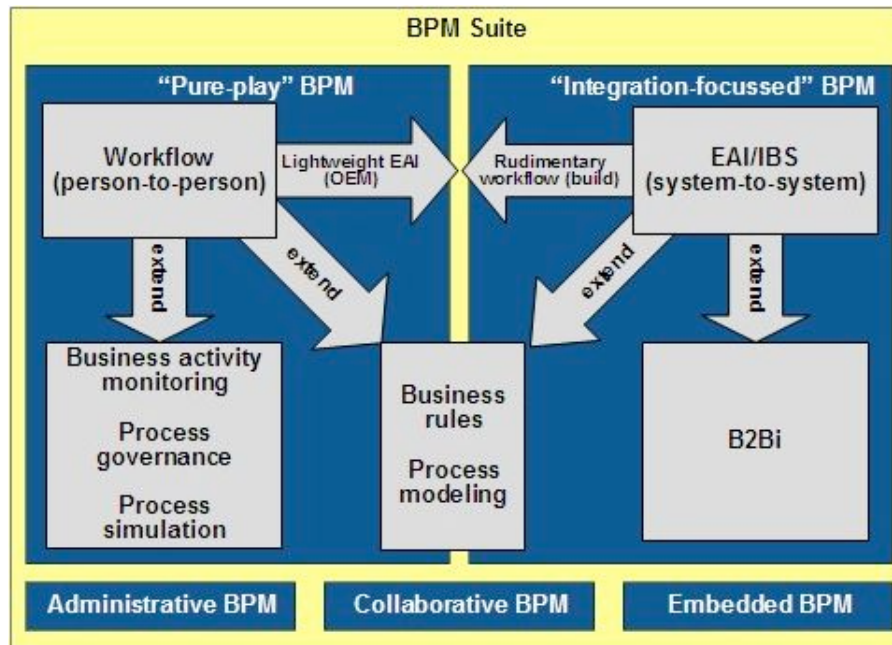


Figure 1.2: Early BPMS Evolution [40]

2. **Integration suites** represents former "integration-centric" BPM suites
3. **DCM suites** represents case management systems

Through market consolidation these three segments are gradually merging into a single BPM suite offering that can cover a subset of or even all three different work patterns: dynamic case management (DCM), human workflow, and straight-through processing. These unified BPM suites now provide single design and development environments for building and deploying end-to-end business processes that incorporate multiple process patterns and use cases.

Moreover, new use cases for BPMS has been still emerging. In addition to the related systems that has been coupled with BPM suites already for some time now (e.g. Business Rules Management Systems, Business Intelligence, or Enterprise Content Management), new rising tendencies appear. An overview of the emerging trends in the BPM area is covered in Gartner's Hype Cycle report [19], [23] (see Figure 1.4). Among these, the following influences appear to be the strongest:

- **Mobile applications integration** - Over the past few years, the way we get work done changes significantly. Increasingly, workers and

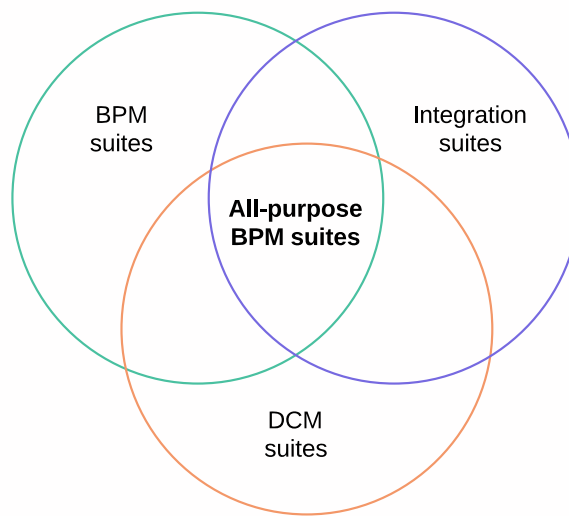


Figure 1.3: Emerging Business Process Management Suites Market

customers expect to engage with companies via mobile applications of all sorts, and BPM suites are not an exception. These technologies provide better level of user interaction flexibility which may lead to another possibility of increasing process efficiency and customer satisfaction.

- **Social channels** - Organizations may utilize enriching BPMS with capabilities well-known from all kinds of social networks. Processes can be identified, designed and iterated in a shared space where each of the process stakeholders may contribute to process improvement. Thus process design becomes more visible and holistic through community collaboration and collective activities. Cooperation can be also used during the operational phase of the BPM project, for example providing guidance of a subject-matter expert to user who is completing a problematic human task.
- **Intelligent Business Operations (IBO)** - Recently emerging set of use cases closely related to the standard usage of BPMS. It is focused on making faster and better decisions in a rapidly changing business context by incorporating new functionalities like real-time business analytics, deep complex-event processing (CEP), social media to support social behaviour and collaboration, and expanded technologies to support growing requirements for mobility. These features intro-

1. INTRODUCTION TO BPM

duce the next generation of BPMS, identified as intelligent business process management systems (iBPMS).

- **BPM Platform as a Service (BPaaS)** - Increasingly, the organizations start to look for outsourcing of business process delivery. With the advent of cloud computing, the business processes may be sourced from the cloud and constructed for multitenancy. Unlike the on-premises BPM solutions, the pricing models for BPaaS are based on actual consumption. That makes it a viable option for a pilot or dynamically growing BPM projects.

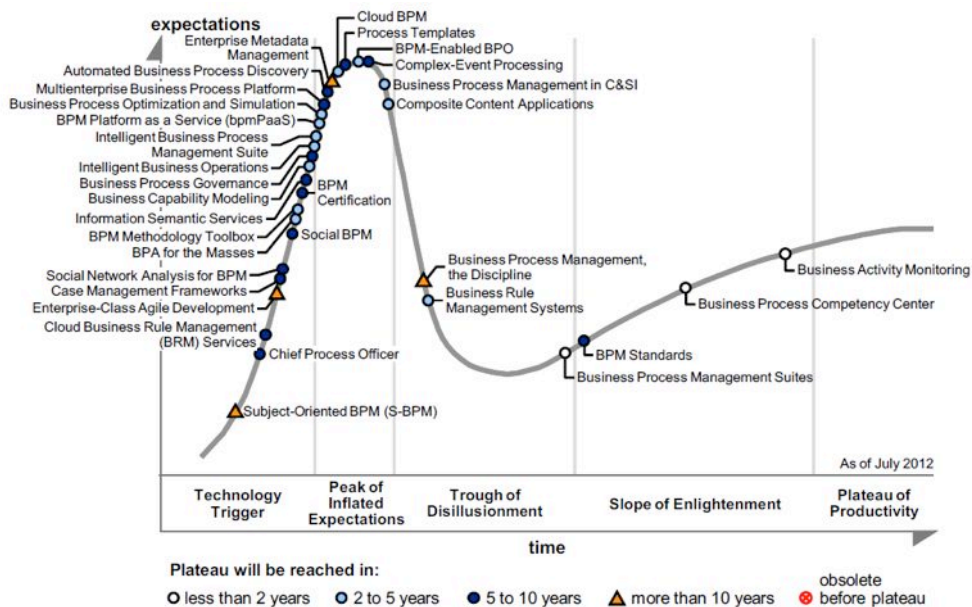


Figure 1.4: Gartner: Hype Cycle for Business Process Management, 2012 [19]

In 2011, the latest version (v2.0) of the BPMN¹ was released. The major innovation consisted of introducing option for seamless, well-documented, straight-forward translation of the business process model into executable process. Many BPMS vendors decided to make use of this capability as soon as possible in order to offer BPMSs that would be more agile in terms of delivering the final business process solution ready to execute in production.

¹for more information see subsection 1.3.4.1

1.3 Characteristics of BPM

1.3.1 Benefits of BPM

BPM is about managing business activities in a comprehensive way. While adopting BPM offers many benefits, the primary motivation for applying it in a given organization may differ. For example, some organizations may be focused on executing their activities more efficiently (that is, producing the same output with less resources like time, money, goods, and labour), while others may be more interested in creating higher business agility in order to respond better to changing market conditions. In some cases, process management may be necessary to produce sufficient visibility and create audit trails across a chain of activities so as to meet a variety of regulatory compliance requirements. Table 1.1 summarizes the key BPM benefit categories, along with some of the typical metrics that can be used to monitor the benefit levels actually achieved.

Table 1.1: Key benefits of BPM, their outcomes and typical metrics (based on [16])

Category	Outcome	Metrics
Efficiency	Lower cost, better resource utilization and productivity, higher product and service quality	Capacity utilization, level of automation throughput, response time, quality versus cost
Visibility	Knowledge of status of business activities and end-to-end transaction, lower capital reserves	Financial and accounting measures, SLA failure rates, risk reduction, rate of non-compliance, cross/up-sell
Agility	Growth in revenue and market share, increased competitiveness, product/service and thought leadership	Speed of new process creation and adjustments to existing processes, time to market with new offering
Business-IT Alignment	Joint ownership of BPM adoption, continuous business process improvement	Level of IT-leveraged business innovation, success rates of IT delivery of business solution

1.3.2 BPM Life Cycle

Most engineering methodologies deal with software development life cycle in a similar way. Typically, it comprises of analysis, design, implementation, and deployment. In modern engineering, these phases are performed iteratively while making incremental changes. The BPM life cycle is not so different from this scenario, though there are some differences though. Primarily, more emphasis is given on monitoring, analysis and optimization that drive the iterative cycles of continuous process improvement. The concept of the BPM is relatively new, the initial model provided by Smith and Fingar in [62] is depicted in Figure 1.5. It comprises of six phases based on Plan-Do-Check-Act¹ principle.

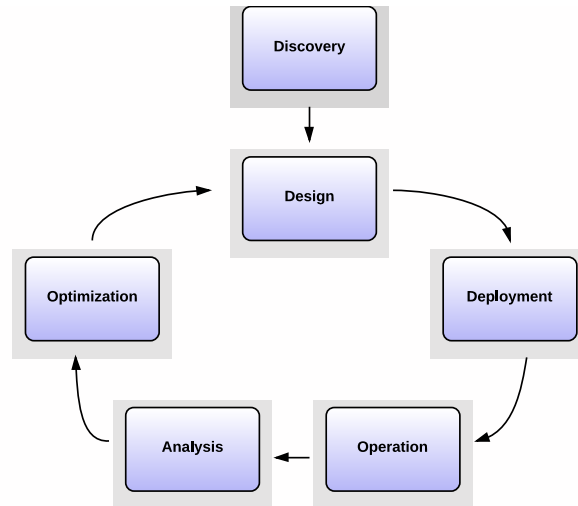


Figure 1.5: BPM Life cycle by Smith and Fingar [62]

Since then, some other authors followed with altering the model and added or skipped steps according to whether the point of view was biased more towards the business or the IT perspective (e.g. [7], [25]; for comparison see [58]). The life cycle model proposed by Koster in [44] appears to be the most appropriate to our cause (see Figure 1.6), as it has been created to reflect the aspects of implementation of processes in BPMS. The stages are described with respect to the cited resources and the authors own experience.

¹(PDCA); it is an iterative method used in modern quality control management, developed by Dr W. Edwards Deming

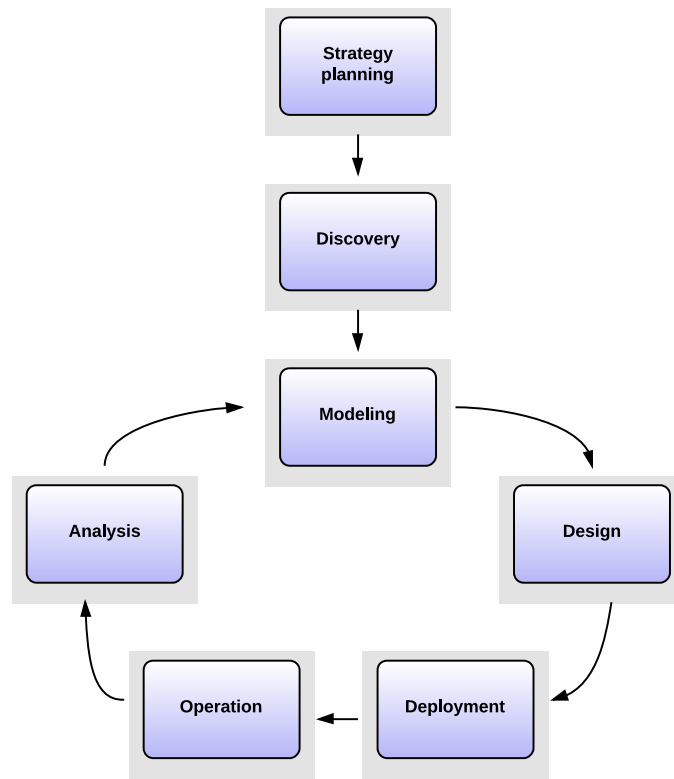


Figure 1.6: BPM life cycle by Koster [44]

1.3.2.1 Life Cycle Overview

First of all, *strategy objectives* of the organization should be defined and linked with the high-level business processes. The following *discovery* phase is performed before the iterative cycle in order to obtain descriptions of individual processes in the form of finer-grained process models. The *modelling* phase represents refining the AS-IS process models and documenting them in a formal modelling language, their validation, initial analysis, and simulation-based optimizations. The design phase follows with translating the model into an executable form, implementing integrations and detailed user interfaces. The process is then ready for deployment which starts the operation phase. As the process instances are started, users follow the activities and the BPMS executes the implemented process flow. To maintain control and provide inputs for the subsequent phases, process data are gathered and monitored. During analysis, the data are subjected to investigation with the help of specialized tools and used for optimization. That includes searching for bottlenecks in the process, elimination of inefficien-

1. INTRODUCTION TO BPM

cies, and other efforts to improve the process.

1.3.2.2 Strategy Planning

Before a BPM project life cycle may start, the strategy development should be performed. The organization management defines the strategic objectives that they will try to achieve and that drive the development of the entire organization. They can be captured by using, for example, a Balanced Scorecard. These identified objectives should be put into the context of business processes of the organization. Thus, high-level overview of the business processes in a form of coarse-grained aggregation (value-added chain diagrams, process maps - see Figure 1.7, etc.) of the business processes need to be created and linked to the objectives.

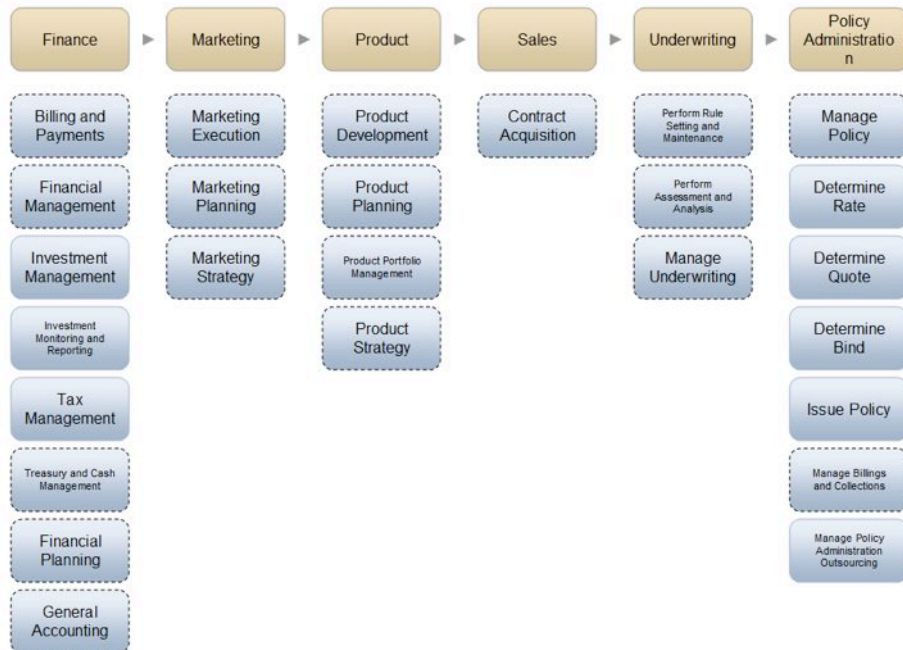


Figure 1.7: Process Map [27]

In order to enter the continuous improvement cycle, an analysis activity is required to identify business processes and select those most appropriate for automation. Since this step can be the determinant of the overall success of the BPM adoption program, it is of capital importance, mainly in the early stages. The main goal is to identify the particular business processes that would be the most suitable to be modelled and implemented using a BPMS. The selection should correspond with the determined

business strategy and focus on the areas that would bring the organization undisputed added value in terms of increased agility, reduced costs, and so on.

Only a thorough evaluation of the candidates for the process implementation may provide foundations of sponsors' trust in BPM principles. The first pilot BPM project should strive to bring the organization the highest value possible in the shortest time. The pilot serves as an evaluation of feasibility, time, scope, and costs prior to performing a full-scale project. Ideally, it helps to increase the management buy-in by proving the beneficial effects of the BPM. [34]

1.3.2.3 Discovery

The process discovery phase consist of acquiring the initial versions of AS-IS¹ state of business process models. These models are created by observing and documenting the way the organization is currently working and gathering all the relative information; the typical means of fulfilling this task are interviews with the process participants, investigating documents and existing software systems supporting the process (if they exist).

The process discovery can be done by two methods:

1. **Manual** (also called **business process mapping**) - Expressing the model as a set or a flow of activities performed by participants (users or systems) by a business analyst. The outcome may have a form of a flowchart, UML activity diagram, Ishikawa diagram, Business Process Model and Notation (BPMN), etc.
2. **Automatic** (also known as **process mining**) - Extracting the business process from existing implementations in information systems. The mining methods use transaction logs to distil a structured process description [1].

The discovered business processes incorporate mutual relationships, such as succession (the output of the first is the input for the second process), that form the process architecture.

During the process discovery phase, the use of tools supporting communication and collaboration among the participating members may be highly beneficial. The modelled processes should be drawn as thoroughly as possible; that means that no significant parts are neglected. Different

¹current state of the business process, before making any change; opposite to TO-BE state

participants might see the modelled processes from a slightly different point of view, so they are able to complement each other's knowledge of the process and to end up with a unified process description.

1.3.2.4 Modelling

The discovery phase provides an output in the form of models captured as an informal model. To be able to perform the subsequent steps, the business processes must be expressed and recorded in a formal manner. The most important characteristic of the model is its unambiguity so any alternative interpretations can be ruled out. This might be supported by representing the same reality from different perspectives using more than one model.

Over the past years, a number of modelling methodologies and notations have been developed, including UML 2 Activity Diagrams (AD), the Business Process Model and Notation (BPMN), Event Driven Process Chains (EDPC), IDEF3, Petri Nets, Role Activity Diagrams (RAD), etc. Each of them was designed for different purposes, therefore the authors focused on the specific perspectives that suited their case the most. We can find more about the modelling methodologies together with their evaluations in [24], [65], [50].

For the purpose of business process automation, I find the BPMN as the most suitable notation. In its version 2.0, it meets the requirements for efficient transition of the AS-IS model into executable business processes (simply because there are no explicit steps required). Moreover, an important aspect of the BPMN is that a single model type can be used that is comprehensive enough for the business stakeholders and at the same time provides a sufficient level of detail and capabilities for the technically oriented people. The BPMN is described in more detail in subsection 1.3.4.

For the modelling purposes, there exists a variety of software tools. The differences are especially in supported notations, conformation to standards, or simulation capabilities. Pure modelling tools do not support the entire BPM life cycle including the process execution and governance which distinguish them from the full-scale BPMSs. An example of business process modelled in software tool (namely BizAgi Process Modeler) is shown in Figure 1.8.

The created model (or more often a set of models) may be used as a basic source for analysis of the business process, discussions about both AS-IS and TO-BE states among the stakeholders and gathering the requirements for the implementation phase.

To achieve an improvement through process implementation, a set of measurable criteria needs to be defined. The strategy objectives form a

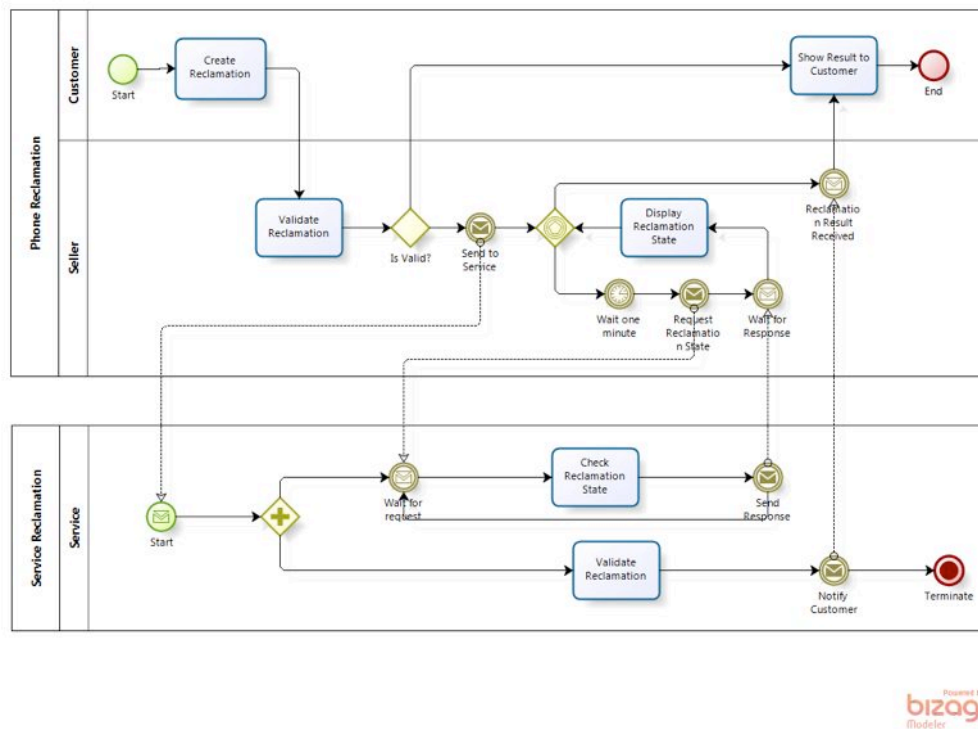


Figure 1.8: Business process modelled in BizAgi Modeler [10]

proper basis for specifying the goals and KPIs¹ that apply for the individual business processes.

A process modelling phase may be a part of an already running life cycle, therefore it follows the preceding iteration. In that case, the current state of all the defined aspects of the process are compared to the previously documented version. The analysis outputs and continuously changing business requirements are incorporated into the redesigned model.

As an integral part of the modelling phase, business rules are usually defined. Business rules describe business knowledge in a formalized way that can be automated. For example, to grant a mortgage, the business rule defines the eligibility conditions. Business rules can either be defined implicitly or explicitly:

- **Implicit business rules** become a part of business process model. Typically, they are defined within a rule task or a decision gateway. When the rules changes, the entire process has to be changed.

¹Key performance indicator

- **Explicit business rules** enable the organization to separate life cycles of business processes and business rules. The process model includes only references to the rules and the rules can thus be stored, managed and modified independently from the process. This increases business agility, especially in a case when the process includes a large number of rules. The rules are stored in and managed by the Business Rules Management System (BRMS) that stands within the enterprise architecture and exposes the rules to be invoked by other applications in the organization.

1.3.2.5 Design

The design phase is usually performed with the support of IT. The main goal is to implement the process¹ which means to build an executable version of the modelled process that would be running on the process engine.

The way of how the executable process is acquired is determined solely by the used BPMS. Therefore, this may be one the strongest criterion for BPM technology platform assessment. Essentially, the automation can be achieved in three ways:

1. **Direct execution of the modelled process.** The latest version of the BPMN (2.0) is designed to enable direct execution of the process model represented in a structured way (e.g. based on XML). That allows much shorter development cycles simply because there is no need for any type of model translation or additional programming. The entire process diagram provides a shared view for both the business and IT stakeholders, so it is always in sync and there is no need to maintain the separate executable version.
2. **Translation of the process from modelling language to executable language.** Prior versions of BPMN (1.2 and lower) as well as many other process modelling languages, do not provide a way how to execute them. They need to be translated into a specialized language designed specifically for such purposes, such as BPEL².
3. **Custom implementation of the executable process.** The executable process is developed from scratch on the basis of process models captured during the modelling phase. The implementation

¹The term *implementation* may be ambiguous. In this thesis, it is used in a sense of a sequence of activities needed to acquire the executable business process.

²Business Process Execution Language

is typically made in a programming language commonly used for enterprise applications like Java or C#. The effort made to undertake the development itself and related software engineering activities are substantially higher than the former two options.

Nonetheless, no matter which of the approaches is chosen, there are activities still remaining to be done during the design phase. Implementation technique of these parts are dependent strictly on the BPMS used.

- **Integration.** The business process in the executable form needs to be enriched with the technical details enabling integration of the process into the enterprise architecture. This includes creation or configuration of adapters to existing systems, databases, portal solutions, or ERPs. The underlying middleware technology plays an important role in the integration. The BPM system can benefit from co-existence of SOA-based technologies like message brokers or ESB. More about SOA-BPM relation in subsection 1.3.3.
- **User interfaces.** User interface (UI) for the process human tasks needs to be developed to provide system-participant interaction. UI design and implementation is either custom-built on traditional web application frameworks like JSF¹ or generated automatically with UI technology embedded in the BPM suite. Some BPM products offer many customization options for building modern rich web interfaces integrated with a business process environment (see example of a user interface design tool in Figure 1.9).
- **User Management.** Defining roles allowed to access different BPMS features and their mapping to actual users is another necessary step. It is convenient to utilize the organization structures contained in the existing systems such as LDAP².
- **SLA.** Targets of service performance are defined to enable monitoring the operation of processes. Thus it is possible to ensure a specified level of quality of executed processes.

After the executable process is implemented, it needs to be properly tested to ensure process quality and mitigate the risk of failures. Testing should be done on different levels - integration, process and user interface. Some of the BPM technologies provide support for testing, sometimes it is necessary to perform the tests manually or utilize third-party tools.

¹JavaServer Faces

²Lightweight Directory Access Protocol

1. INTRODUCTION TO BPM

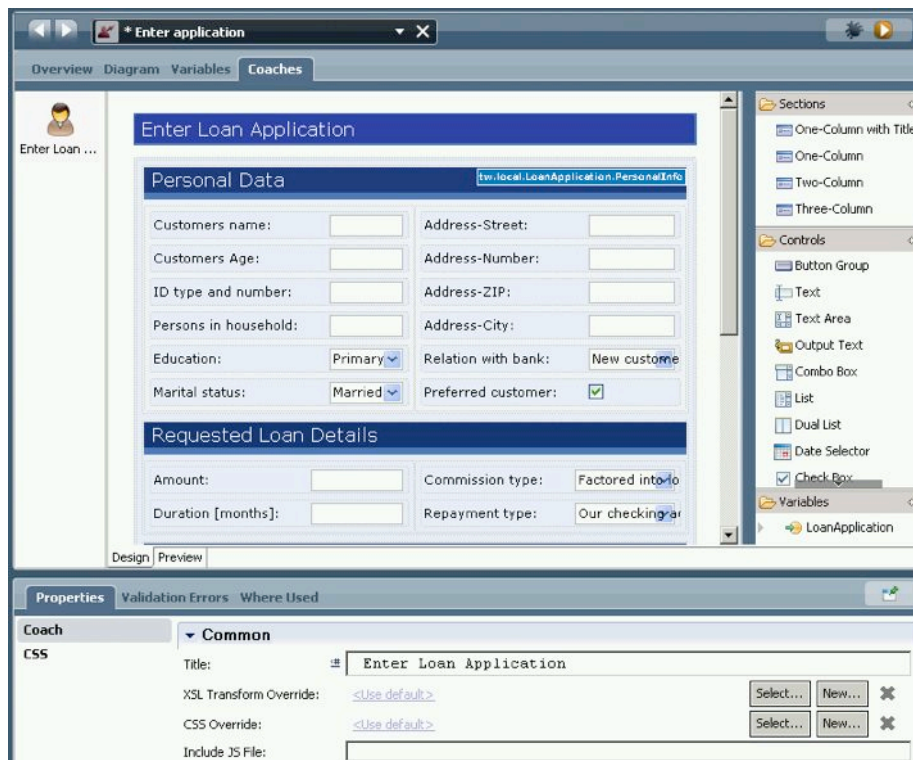


Figure 1.9: User interface design example (IBM Business Process Manager)

1.3.2.6 Deployment

During the deployment phase, the executable business processes are rolled out to the production environment. This may include copying the new versions of compiled processes to the technical infrastructure such as application servers, web servers, business process engines, etc. On a large scale of process applications, this might mean deployment of the process to the distributed environment - server clusters or even multiple process engines spun across multiple organization units.

1.3.2.7 Operation

The operation phase consists of three sub-phases, namely execution of the business processes, interaction between users and the business processes, and monitoring and control of the business processes and their execution environment.

During the execution sub-phase, every time an executable business process model is executed a process instance is created. The process engine

is responsible for the efficient processing according to the modelled process flow. In the course of execution, the process-related data are continuously logged for the process analysis. These data include information about activities, users, routing, events and time of execution. Another thing that is to be handled is **routing** of the process activities, that is, deciding who needs to accomplish the given task. The routing can be done based on roles, business rules, priorities, events, or even the current workload of the participants.

The interaction sub-phase consists of user interactions with the running process. The user usually accesses the process task through a portal application. (see example in Figure 1.10). Its main purpose is to provide a user with list of tasks that he may manipulate. Individual tasks are usually web-based forms that enable the user to read and modify business data.

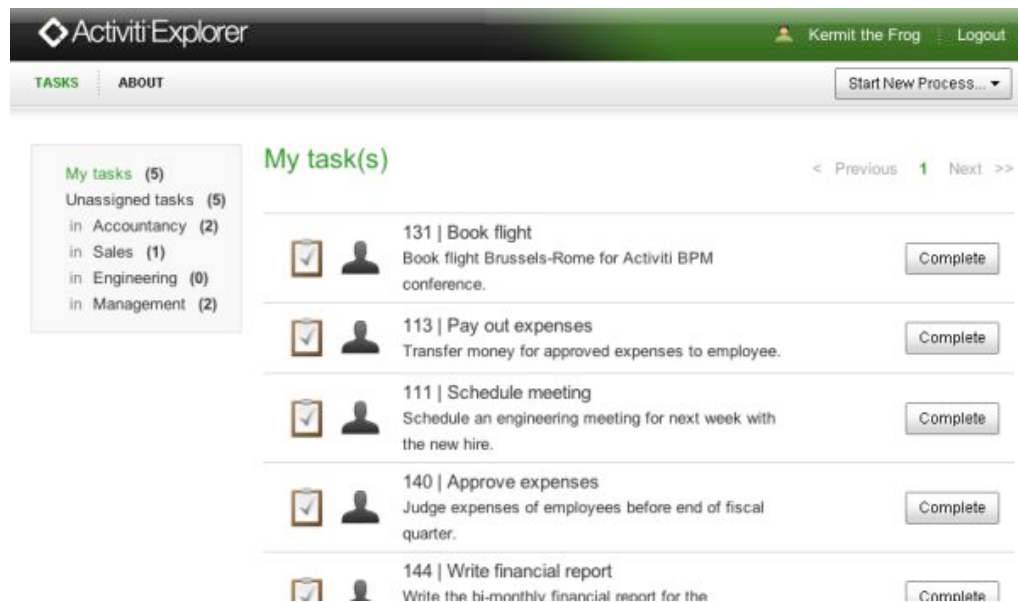


Figure 1.10: Example of user task list in portal (Activiti)

The last sub-phase concerns monitoring and control. That includes managing possibly thousands of process instances running at the same time by specified roles. BPM suites usually provide tools for managing the process instances in real-time and deal with both technical and business issues that occur during the process execution. These tools are called Business Activity Monitoring (BAM).

The BAM monitors the process-related data and the log files of the BPM product and calculates the KPIs of the business process. The gathered performance metrics are evaluated, compared to the bounds defined during the

1. INTRODUCTION TO BPM

modelling phase and, if an exceptional state arises, an appropriate handling event is triggered.

However, the most important part of BAM is the monitoring dashboard, which allows the monitoring of the business process performance. For this purpose, the BAM usually provides a web-based user interface supporting a high-level of customization according to the specific users' needs (see Figure 1.11 for an example). It uses aggregated views of running process instance data in a form of charts, graphs or lists. These dashboards act as a fundamental tool for efficient management of the running process instances. The level of detail provided depends on the target information recipient. Some users want to see only a high-level overview while others might want to drill-down to more detailed information. Furthermore, the BAM may allow the user to manipulate the instances – for example, to terminate the redundant process instance, to reassign the task to another user to balance the workload or to change the priority of a task to speed up the process flow.

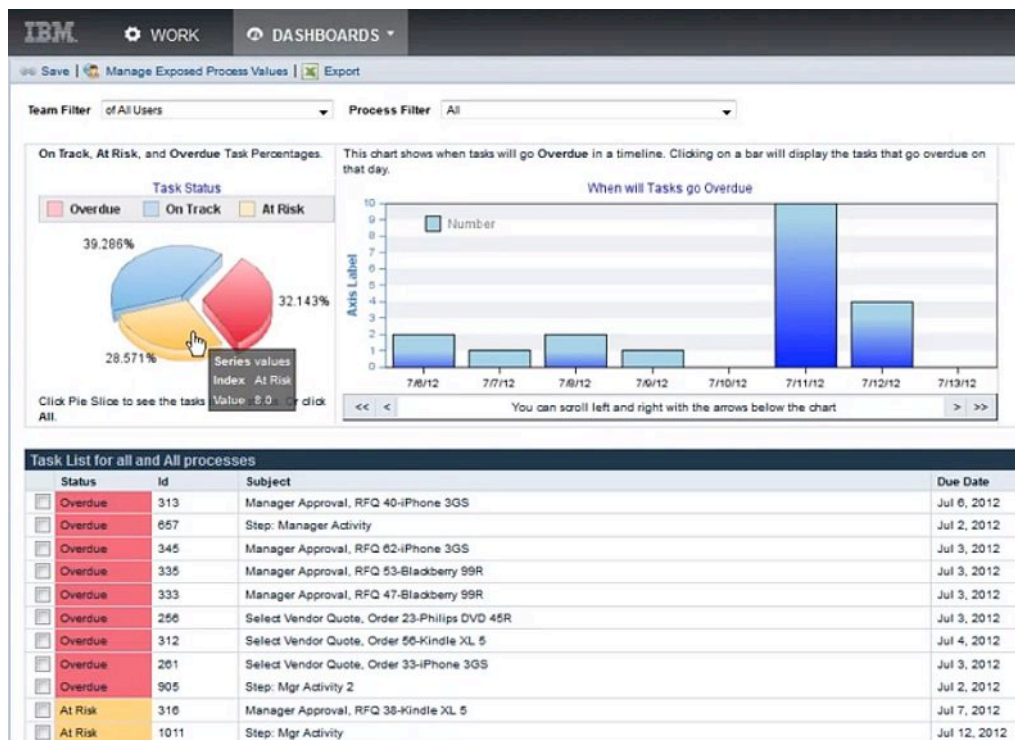


Figure 1.11: Business Activity Monitoring dashboard (IBM BPM)

1.3.2.8 Analysis

Although analysis techniques can be used during any other phase in the BPM cycle, they are mostly used before incorporating new changes during the process modelling and design phase. The analysis findings may be very useful for making the process change significantly more efficient.

The analysis includes various methods such as verification, simulation, historical data analysis or optimization. Verification consists of checking of the correctness of the model according the common understanding by the process participants, and conformation to the semantics of the modelling language.

Simulation can be used to find out whether the modelled process behaves as expected even before the process is actually run in the production environment. This includes meeting requirements for throughput times, resource utilization, and so on. By contrast, in historical data analysis, the data gathered during the operational phase are investigated. The Business Intelligence (BI) techniques and software tools are often utilized for this purpose. The outcomes may serve as an input to the process optimization or just for reporting to the business owners or executives. The process optimization consists of making modifications in the process model or design according to the identified inefficiencies, bottlenecks, changes in business environment or goals. After the process model changes proposal, a new simulation are performed and its outputs are then compared with the historical data. This procedure verifies the proposed process change to be beneficial in terms of the delivered improvement.

1.3.3 BPM and SOA

Since natural way of using BPM is built up on orchestration of services, it is very often designed and implemented together with an architectural strategy that is suitable to serve as an underlying layer providing infrastructure of resources. SOA may serve as such layer.

1.3.3.1 SOA

Service-oriented architecture (SOA) is *"a business operations strategy for leveraging information to meet the organization's objectives, such as increasing overall revenue, boosting customer satisfaction, improving product quality, and enhancing operational agility"* [20].

With SOA, it is possible to link the resources on demand. These resources need to be available for consumption to participants across the

enterprise. The resources are represented by a set of services specified by business and provided by IT that, when put together, fulfil the organization's business goals. It is possible to choreograph the services into composite applications and the invocation of the service is performed via standard protocols (e.g. SOAP, HTTP). [5]

A service is a (discoverable) software resource with an externalized service description. This service description is available for searching, binding, and invocation by a service consumer. The service provider realizes the service description implementation and also delivers the quality of service requirements to the service consumer. Services should ideally be governed by declarative policies and thus support a dynamically re-configurable architectural style.

SOA can provide the organization an increased level of business agility. It is gained by IT systems that are flexible, primarily by separation of interface, implementation, and binding offered by a SOA. This way the provider of the service does not have to be chosen until a given point in time which is based on current business requirements. The requirements for the service may be functional and non-functional (e.g. performance, reliability, security, etc.). [5]

1.3.3.2 Elements of SOA

SOA is based on four key abstractions: *application frontend*, *service*, *service repository*, and *service bus* (see Figure 1.12). Services provide business functionality that the application frontends and other services can use. A service consists of an implementation that provides business logic and data, a service contract that specifies the functionality, usage, and constraints for a client of the service, and a service interface that physically exposes the functionality. The service repository stores the service contracts of the individual services of a SOA, and the service bus (such as Enterprise service bus (ESB)) interconnects the application frontends and services.

1.3.3.3 SOA Key Principles

An SOA application adheres to the following five principles [2]:

1. **Modular:** The system has a number of components (usually tens), including at least one component that acts as a service consumer and another that acts as a service provider. Complex problem is then divided and solved by a set of small components that work together.

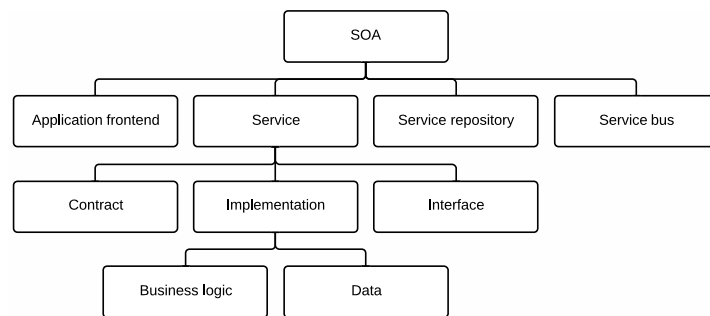


Figure 1.12: SOA Elements

2. **Distributable:** The components can run on disparate computers and can communicate with each other by sending messages over a network at runtime. SOA relies on program-to-program communication.
3. **Defined interfaces:** Component interfaces are documented using metadata that specify an explicit contract between consumers and providers. This metadata describe the messages that are exchanged and other characteristics of the agreement among the components (e.g. Web Services Description Language (WSDL)).
4. **Loosely coupled:** A provider component can be swapped out for another component that supplies the same service without changing or recompiling the consumer (or consumers), because the interface is separated from the service provider's implementation (the provider component's internal code and data).
5. **Shareable:** A service provider component are designed and deployed in a way that it can be used successively by disparate consumer components (sometimes called "reuse").

These characteristics forms foundations that enable the entire IT ecosystem to be well-scalable on long-term basis and thus able to satisfy strategy needs for the expansion of the organization. Moreover, it can reduce the time, effort and cost needed to implement or change distributed application systems compared with other approaches [60].

1.3.3.4 BPM and SOA Synergy

While BPM and SOA each are of value on their own, adopting both may bring the organization a natural synergy they possess when done together.

1. INTRODUCTION TO BPM

BPM provides the business context, understanding and metrics, and SOA provides a governed library of well-architected service and information building blocks. Such BPM and SOA symbiosis is depicted in Figure 1.13.



Figure 1.13: Symbiosis of BPM and SOA [37]

In the typical context of BPM and SOA, one can say that "processes run on services," therefore (automated) activities in orchestrated processes consume services as a part of their execution. Thus BPM is dependent on SOA. Yet the reverse is also true, as in many cases the embedded business processes are a part of the realization of the capabilities provided by a higher level service. Behara stated in [9], that by using BPM, SOA is tied to the process services to develop composite business flows. BPM adds additional runtime power for service composition and the ability to modify a flow in exchange for more runtime complexity. BPM can also provide the assurance that long-running processes are performed and run any necessary compensating transactions in the case of failure. BPM leverages and extends SOA's power by adding a flexible, agile runtime layer to the services exposed by SOA.

Eventually, the decision to build up BPM and SOA together or not is always a complex task. Apart from the already stated advantages of doing so, there might be possible negative consequences if one of the parts is left out (as Jensen et al. stated in [37]). That should be taken into account by a potential adopter. Some of them are listed in Table 1.2.

Implementing BPM and SOA together has one beneficial side effect. Visual representations of business process models and service orchestrations facilitate communication and collaboration across business-IT borders. If the business contracts are defined explicitly, mutual trust increases which results in better linkage consequences between business units and the realization of end-to-end processes.

Table 1.2: Possible negative consequences of adopting BPM and SOA apart (based on [37])

SOA without BPM	BPM without SOA
<ul style="list-style-type: none"> • Lack of a disciplined approach to creating and managing an agile library of well-architected and reusable building blocks(including all of services, processes and information assets) • Lack of governance and lack of explicit contracts between business and IT participants in an end-to-end process • Lack of context for optimizing investment across business and IT 	<ul style="list-style-type: none"> • Lack of disciplined approach to process definition and optimization • Lack of context for business operational excellence and for managing business operational risk • Lack of explicit metrics for the business value of service reuse

1.3.4 Standards Used in BPM

Increased expansion of BPM practice has led to the need for establishment of norms commonly used in the BPM domain. Since the development of various BPM trends, methodologies or software products still remain rather active, the standardization activities are introduced to prevent fragmentation. Even though the BPM domain does not conform to any universal set of norms, there exists certain specific *de facto* standards accepted by a considerable number of practitioners.

The standards are generally a very important aspect for the potential adopters of the technology. The proprietary solutions may cause them to hesitate before entering the field because they do not want to end up tied to a specific technology preventing the uncomplicated change of the vendor.

In this thesis, which focuses mainly on executable aspects of the business process management, the BPM stands as the primary and most important standard because of its recently gained applicability to the process execution.

1.3.4.1 BPMN

The Business Process Model and Notation (BPMN) is an industry standard defining a graphical notation language designed for representation of business process diagrams. It was developed by the Business Process Management Initiative (BPMI) and is now maintained by the Object Management Group (OMG) since the merger in 2005. The main objective of the notation, as defined by the OMG, is to be *"readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes"* [53]. Thus, the BPMN is positioned at the interface between business and IT, as depicted in Figure 1.14.

At first glance, one might say that UML¹ activity diagrams cover a very similar scope of capabilities to model business processes. Although both modelling notations provide a representation independent from the implementation, the BPMN is more convenient for both key aspects of business process modelling, that is, it facilitates discussions with the business stakeholders about the scope and functionality, and also it is designed for straightforward execution by a BPM engine.

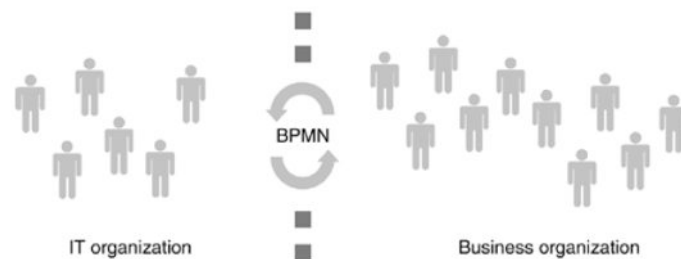


Figure 1.14: BPMN forms the interface between business and IT. [45]

BPMN can be used for two main modelling purposes. Firstly, it can convey the process flow as it exists within the organization. This process model AS-IS serves mainly for documentation and analysis. The processes are usually captured in a top-down approach, that is, the modelling begins with the high-level activities and continues with more detailed levels. The level of captured detail depends on the particular audience and purpose. The second application is for designing the TO-BE state. That represents

¹Unified Modelling Language

the optimized version of the process that is intended for execution. Besides covering the business activities and the flow it includes all the technical details necessary for deployment to a process engine.

The BPMN specification itself does not contain any guidelines or conventions how to model processes to make the best use of them. However, some literature has been published to help analysts with the modelling practice. Bruce Silver in his book called "BPMN Method and Style" [61] goes beyond the BPMN specification and presents the guidelines that can serve as a starting point to establish the organization's own modelling methodology.

Elements set

In BPMN v2.0, business processes can be represented by three different sub-models - Processes, Choreographies and Collaborations. However, for an end-to-end process representation that corresponds to the process execution flow, the Process sub-model is used. It is modelled using a set of graphical symbols that have their special meanings. They are grouped in the following way:

- **Flow Objects** define the behaviour of the process. They include:
 - *Activities*, are repeating actions with defined inputs and outputs performed by users or system.
 - *Events*, are used for handling signals in a flow. There may be starting/ending events, time triggers, message events and many others.
 - *Gateways*, which provide a way to split and merge the process flow. They define conditions that specify how the flow is controlled.
- **Data** represent data objects related to the process.
- **Connecting Objects** connect flow objects to each other.
- **Swimlanes** are used for grouping of the elements.
- **Artifacts** provide additional information about the process.

The core elements are depicted in Figure 1.15. The full specification of the BPMN can be found at [53].

Process execution with BPMN 2.0

Although BPMN had been standardized and widely used by business analysts for process modelling, its original purpose was not to enable direct execution of the captured process. This fact became very important when the organization wanted to move to the next level and automate the process. The BPMN models needed to be transformed to a process execution language (such as WS-BPEL¹) that would enable to interpret the model by a process engine. The problems with this approach were several. Firstly, due to lacking constructs in the specialized execution languages, the conversion was never possible to do exactly one-to-one. Secondly, the implementation cycle was extended which decreased the agility of business. And most importantly, it meant to think about one thing in two different ways that lead to misalignments between business and technical oriented people. [56]

That was changed by introducing BPMN 2.0 which was defined as a standard for both process modelling and implementing an execution model. It brought an opportunity for the two sides to speak with the same language. In addition, it encouraged the BPMS vendors to redesign the suites to make the implementation of business processes significantly simplified.

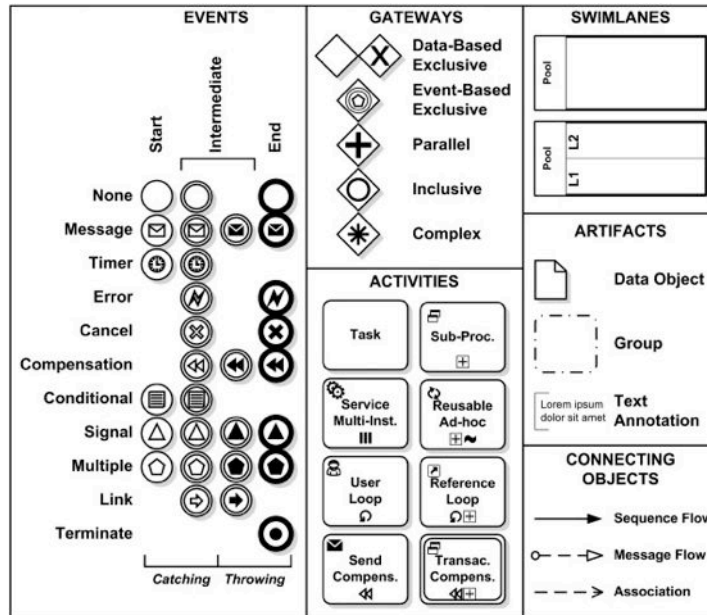


Figure 1.15: Core BPMN elements (in BPMN v1.2) [13]

¹Web Services Business Process Execution Language

1.4 Summary

Business Process Management creates conditions for improving efficiency, visibility, agility and business-IT alignment in the organization. Although its principles have been studied and used for a long time, the complete software solutions supporting the entire life cycle of the BPM program had not emerged until a few years ago. Currently, Business Process Management Suites represent a viable opportunity for improving business processes through automation while gaining positive synergies from SOA. Along with the technology, the BPMN standard has been developed to encourage the potential adopters to start the BPM initiatives more easily.

Reusability in BPM

2.1 Introduction to Reuse Principles and Approaches

Software reuse has always been one of the most important principles of software engineering. By means of reuse, software development in general has become more agile, less expensive and it makes the software developer's work process more convenient in terms of that he can focus more on the solution design instead of mechanically repeating the same steps.

There exist many different viewpoints of what the software reuse is. In our context, we adopt Kruger's [46] general view that defines it as follows: *"Software reuse is the process of creating software systems from existing software rather than building them from scratch."*

Then, reusability can be defined as *"a property of a software asset that indicates its probability of reuse"*, as stated by Frakes and King in their extensive survey [22], where they also provided an overview of significant literature related to software reuse.

Software reuse has gained increasing attention throughout the history, alongside with the extent of software development practice. In the 1960s, when the field of software engineering is considered to have been established, the software developers attempted to face the problem of building large-scaled systems in a cost-effective way by reusing software for the first time. These first attempts consisted of creating a library of reusable components that could be reused repeatedly in different situations [46]. As the software systems that were created grew in size and scope, software reuse principles were applied on increasingly coarse-grained entities (see Figure 2.1). Therefore, recent software engineering initiatives have been us-

2. REUSABILITY IN BPM

ing different approaches, such as Model Driven Development, Asset-Based Development, Feature-Based Approaches, Software Product Lines, and so forth, to improve the reuse. All of them have been aiming to improve systematic reuse with different perspectives to compose very complex systems out of pre-built components. [14]

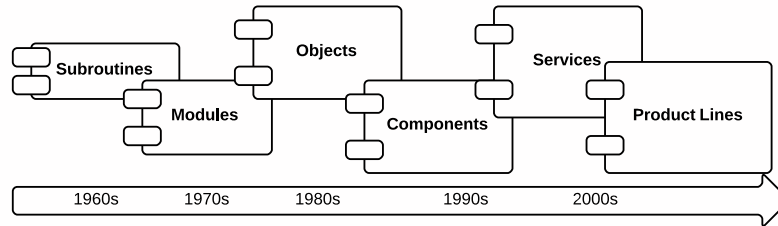


Figure 2.1: Approaches to software reuse in time (based on [14])

2.1.1 Reuse Motivation

Reusing software components is not always an easy thing to do. Although there are some non-trivial obstacles that stand in the way of practising the reuse, the predominating motivating factor is utilizing the work that had already been done. Sametinger in [59] summarized the benefits of reusing collected from literature into the following groups:

- **Quality Improvements**

- *quality* (reuse results in a decreased number of errors in the asset)
- *productivity* (time savings for analysis, design, and coding)
- *performance* (reuse promotes optimizations)
- *reliability* (reuse increase chance to detect errors)
- *interoperability* (consistent implementation of interfaces)

- **Effort Reduction**

- *redundant work, development time* (less effort made due to avoiding repeated development)
- *time to market* (reusable components are ready to use)
- *documentation* (along with the components, the documentation is reused)

- *maintenance costs* (fewer defects from components of quality)
- *training costs* (gaining knowledge from using the components)
- *team size* (when reusing, the teams can be smaller which increases productivity)

- **Other Benefits**

- *rapid prototyping support* (reusable components enable faster creation of prototypes)
- *expertise sharing* (studying implementation of the assets may lead to increased design skills)

2.1.2 Reuse Taxonomy

Prieto-Díaz in [55] identified six basic perspectives from which to view software reuse. The facets of reuse are summarized in Table 2.1 together with a short descriptions and examples.

Facet	Description	Examples
Substance	nature of the reused items	ideas, concepts, artifacts, components, procedures, skills
Scope	extent of the reuse	vertical, horizontal
Mode	how the reuse is conducted	planned, systematic, ad-hoc, opportunistic
Technique	approach to reuse implementation	compositional, generative
Intention	ways to apply the reusable items	black-box, as-is, white-box, modified
Product	subject of the reuse	source code, design, specification, objects, text, architectures

Table 2.1: Facets of reuse by Prieto-Díaz [55]

This taxonomy and its parts can give us a good overview in various aspects of the software reuse. In this thesis, the main focus is reuse in the context of BPM application development, so these aspects can serve as the basis for categorization of assets created during the BPM projects.

2.2 Asset-Based Development

The topic of this thesis is linking Business Process Management and software reuse. In this section, Asset-Based Development, which is the best practice-based approach for reusing, is described as an approach suitable for putting BPM development into relation with software reuse practices.

The Asset-Based Development (ABD) was created almost a decade ago by a consortium of software industry leaders – including Rational Software (before being acquired by IBM), IBM, and Microsoft. The main goal was to deal with the challenges of improving the return of investment and overall quality in software development. The creators concluded that it can be achieved by reusing solutions that had already been developed by applying appropriate methods for reuse practice like naming, organizing, reviewing and using the parts of the software [48]. This subsection is based on knowledge contained in the following publications that elaborate on ABD [18], [48], [52] and [47].

According to DeCarlo in [18], the Asset-Based Development is *“developing software solutions reusing cohesive, documented software artifacts. It is organizing software development in a way that leverages previous investments, and influences the nature of future investments. It is speeding up development, and reducing cost and risk by reusing assets and artifacts to solve recurring problems.”*

The cornerstone of ABD is an asset (depicted in Figure 2.2). It is defined as *“a collection of related artifacts that provides a solution to a problem. The asset is customizable through its variability points, meaning those locations within the asset (or more specifically, within the asset’s artifacts) that can be customized.”*

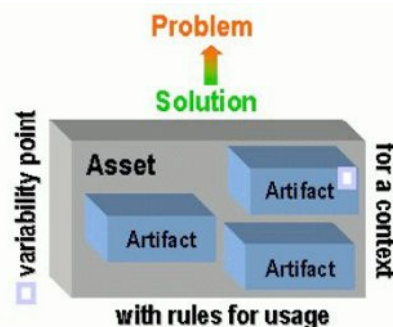


Figure 2.2: An asset as a collection of artifacts [52]

An asset may have different forms. It may be an integrated solution

to a problem containing requirements, use cases, models, designs, source code, specifications, templates, test cases or even data. Alternatively, it can be represented by a single artifact ready to use in AS-IS form or just a guideline or a textual description of solution to a recurring problem.

2.2.1 Asset Life Cycle

Figure 2.3 illustrates the asset life cycle. While an asset goes through its life, its shape, status and quality changes. At the beginning, it needs to be *identified*, and then the production of the asset can be started. *Asset production* includes gathering all the required artifacts (elements) to compose the new asset and the asset packaging. Before it becomes a part of the asset repository, it must usually go through a revision and/or validation. By *asset management* we mean taking care of the existing assets; it supports their storing, configuration, or removing from the repository. *Consuming* an asset usually precedes searching, and it is followed by providing feedback by the Consumer back to the Asset Manager and the Producer.

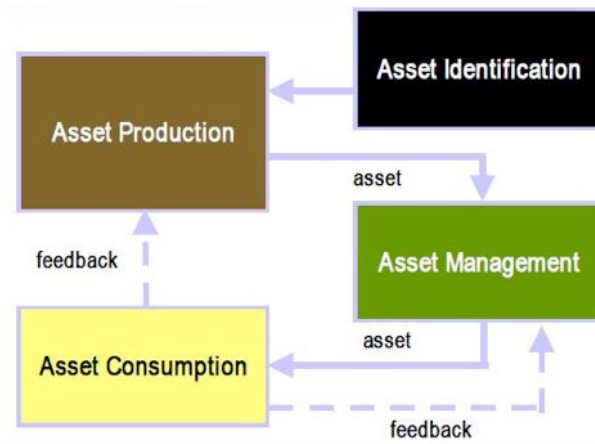


Figure 2.3: Asset life cycle and governance [47]

Throughout its life cycle, the asset is a subject of following use cases (depicted in Figure 2.4):

- **Harvest Asset** (Performed by an experienced designer/architect) - Decides the content and variability of the asset.
- **Package Asset** (Performed by an individual with writing skills and some knowledge of installation procedures) - Creates RAS description of the asset. The delivery mechanism needs to be considered.

- **Catalog Asset**
Decides on how to classify the asset and puts the asset into the catalog.
- **Purchase/Deliver Asset** (Performed between the Broker and Consumer)
Performs the transaction of purchasing assets and delivering them to the Consumer.
- **Search Asset** (Performed by the Consumer)
Identifies potentially usable assets.
- **Analyze Fit** (Performed by the Consumer)
Analyzes the asset description to decide if it can be reused and how. Assumes access to most of the asset description.
- **Apply Asset** (Performed by the Consumer)
Applies the asset to the project.

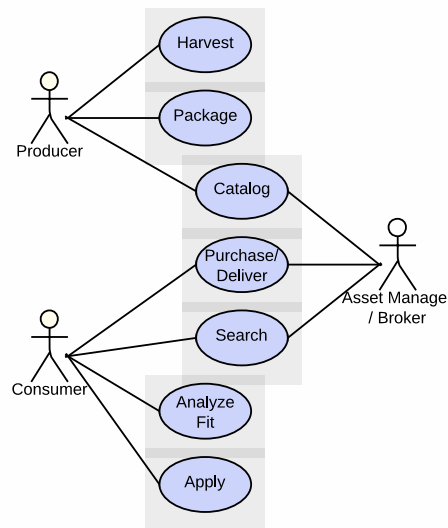


Figure 2.4: Asset use cases (based on [18])

2.2.1.1 Asset Creation

The **asset creation** consists of several steps:

Firstly, the potential asset is ***found***. It is identified, either by realizing a commonly recurring problem or recognizing a solution in an existing application that might be generalized. The asset is then examined for its potential – whether good asset rules are applicable, what might be some

possible side effects, and what's the rationale. Also, the scope of the asset needs to be determined. At this point, the decision whether to build the asset or not should be made.

Second, the asset *extraction* starts with copying the required pieces into new project. All ends need to be cut and modified in a way that enables to use the asset on its own. Variability points¹ of the asset need to be defined and documented.

Third, the *generalization* of the asset follows. The asset should not contain any signs of the original domain. Therefore, all the references to the domain specific terms that are not in the scope need to be removed. The suitable options provided by the asset should be parametrized. The more options the asset provides, the higher level of consumability it possesses.

Eventually, the asset needs to be properly *tested*. All the modifications need to be covered by the test cases. Common principles applied for software testing should be adhered to. Apparently, the testing of the reusable asset is even more important than it is for a non-reusable software component. Every defect would affect all the applications that use the asset.

2.2.1.2 Searching the Asset

The number of the assets created and available for reuse may grow very fast over time. The reusable assets pool may contain thousands of items, and that brings the searchability aspect of the asset packages to focus. The assets are usually stored in a repository. One of the key features that the repository includes is a way of efficient asset search. To leverage the reusability initiatives at the highest level, it should be possible to search by all the metadata defined in the asset manifest. Filtering and sorting based on given criteria may positively affect the search results, so the Consumer finds the right asset that suits his requirements. The search engine interface can be designed as an electronic form, as a search field with possibility to apply syntax of a query language designed for the purpose, and so on. See example of interface for asset search in Figure 2.5 (taken from [18]).

2.2.1.3 Applying the Asset

To successfully apply an existing asset, the following steps are need to be performed. First of all, the Consumer reads through the asset's documentation including the guidance contained in the usage section. When he understands the asset sufficiently, he executes all the installation steps

¹Variability point is a location in the asset that may have a value provided or customized by the asset consumer

2. REUSABILITY IN BPM

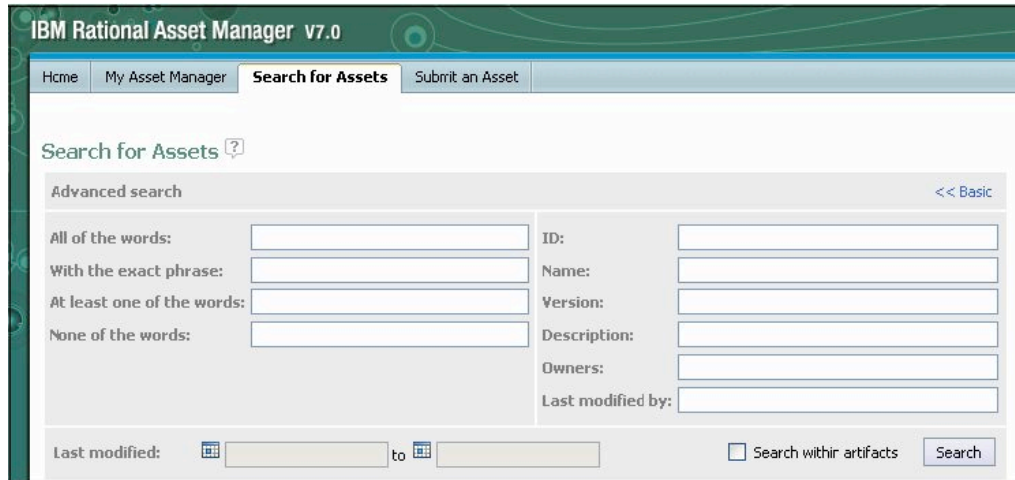


Figure 2.5: Search for assets using advanced search fields [18]

required. If the asset contains any variability points, the concrete elements have to be provided. At this point, the asset is ready to be tested using the test scripts provided by the Producer. Eventually, it may be beneficial to make the applied asset available to the target asset consumers who can start to build their solutions by utilizing the asset. An optional, yet a very important step is to give feedback to the Producer of the asset specifying the asset, overall usability and any experienced problems.

2.2.2 Reusable Asset Specification

Although the Reusable Asset Specification (RAS) was submitted back in 2005 and has not been updated since, it provides a suitable solution framework to the complementary issues within the software reuse. It is not an approach to the creation, design or implementation of reusable artifacts. Instead, it is related to the reusability techniques from the operational point of view. It addresses problems like searching, storing, understanding, organizing, and applying solutions within the given platform.

According to the Object Management Group (OMG), the standards consortium that submitted the RAS specification [52], the RAS is "*a set of guidelines and recommendations about the structure, content, and descriptions of reusable software assets*". It covers generalized representation of different asset types.

Among the properties of a good asset we can count the following (according to [18]):

- **Purpose easy to understand.** The motivation to use and key requirements and constraints are well described.
- **Common software engineering reusability aspects.** The asset implementation conforms to principles like loose coupling, high cohesion, sufficient capabilities, or completeness.
- **Ease of application.** The asset should not require significant redesign or modifications. Steps needed for incorporating the asset into a created solution should be simple, accurate and clear.

To achieve these goals the asset should be enriched with additional information supporting the artifacts within the core asset, such an explanation of the original goal and motivation, models that visualize the artifacts or their relations. Also, providing several examples of possible usage is advisable. All the information added improve consumability of the asset. If the assets are not enriched in this way, the effort invested into creating the asset is wasted.

RAS provides a way of asset description using a XML manifest document. This metadata containing structure is added to the asset's packaging that is specific for the developed application platform. The metadata document conforms to the RAS Metadata Format represented by an XML Schema. The RAS packaging contains the following items (diagram in Figure 2.6):

- **Classification** defined in a form of name and value descriptors, tags, or deployment context.
- **Usage** represented by instructions on applying and customizing the asset.
- **Solution** contains the actual core asset artifacts.
- **Related Assets** define relationships to other asset and help the consumer create collections and larger solutions.

2.2.3 Asset Repositories

When applying reuse within an organization, it is advisable to leverage suitable software tools that offer capabilities to automate and manage the reuse initiatives. They are used to support the entire life cycle of a set of reusable assets. Besides the basic features that would be expected from the this type of software (like storing and providing access to the assets), the asset repositories often support some of the following (based on [3]):

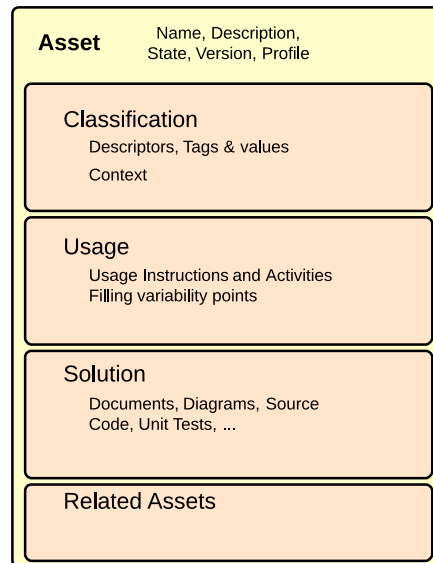


Figure 2.6: RAS Metadata Structure (based on [52])

- **Versioning.** The support for different versions of the same asset is usually provided. Versioning is essential for managing the asset life cycle and its interdependencies within the repository.
- **Integration with integrated developer environments.** Accessing the repository directly from the tool used for the design or development may make the reuse practice considerably more efficient in contrast to using two different interfaces.
- **Search and meta-data.** The asset package providing additional information to the core artifacts can improve consumability of the asset.
- **Discussion capabilities.** The possibility to discuss the issues related to the specific assets brings the repository to provide feedback and thus sustain asset quality improvement.
- **Support for user communities.** The teams with a specific interest within the organization can form communities developing the assets. These communities might have special needs for the asset repository.
- **Asset management workflows.** Like any other subject of software engineering, the assets need to be set into some sort of workflow sup-

porting the quality assurance. That includes, for example, reviewing the assets before allowing them to the target consumers.

- **Integration with configuration management system.** The asset artifact sources may be accessed through standard means of configuration management in support of managing issues and change requests.

The asset repository may have different forms. The scope of the product used for the asset reuse differs according to the organization size, determined approach to the reuse or the reuse maturity.

The simplest solution the may serve as an asset repository is a plain **shared drive** that can be accessed by the reuse practitioners in a rather unmanaged manner. However, this solution cannot bring much benefits to other than really small-sized teams working on short-term project. Another non-specialized solution may be leveraging collaboration solutions as **Wiki pages** or social networks. Recently, these types of software gained a lot of attention and development, and even enterprises started to put them in the centre of their efforts to promote collaboration. Wiki pages are a good option to encourage the exchange information like software assets but without following the standard way to organize this information, and without advanced features that may be offered by specialized asset repository tools.

Another form of asset repository is a **standalone software** tool specially designed for the purpose (to name a few, IBM Rational Asset Manager [31], Atego Asset Library [8], etc.). This type of solution can provide many of the functions described above while being development platform independent and therefore enabling promoting the ABD across the entire organization. However, the costs for purchasing and deployment into the organization run high. The last but not least option is to run the asset repository tools as a part of configuration management system, with user interface **integrated** into an IDE or a modelling/design/requirements management tool (or as a kind of pluggable component). This may be most suitable in terms of efficiency of usage and respecting the specifics of the assets created.

2.3 Measurements and Metrics

To achieve a successful adoption of a new strategy or technology (software reuse in this case), the organization performs activities that were determined for this purpose. However, the outcome of such efforts needs to be measured and convenient metrics need to be derived and evaluated in some way, otherwise it is never clear if the goal was reached or not [15].

2.3.1 Reuse Efficiency

In software reuse, there may be defined various metrics depending on the specific reuse approach and granularity. In this thesis, the main focus lies on achieving the productivity through creating the asset as reusable as possible. Therefore, the goal is to satisfy the following relationship, mentioned by DeCarlo et al. in [18]:

$$\frac{\text{productivity cost}}{\text{using reusable asset}} + \frac{\text{asset production}}{\text{cost}} < \frac{\text{productivity cost}}{\text{without reuse}} \quad (2.1)$$

The idea behind this formula is very simple. It says that to achieve an efficient reuse, the costs spent on using the assets that are created in a reusable manner should be lower than using the assets implemented in a non-reusable way ("from scratch"). Moreover, this fact should not be changed even if the cost for creating the reusable asset is added to the costs for its use, no matter how expensive it is. Apparently, this property depends on how many times the asset is actually used after being created.

Software Productivity Consortium in [15] described a Reuse Capability Model that was designed to help organizations to self-assess and improve their reuse capability. Within the model, they proposed a set of measurements to determine the ability of an organization to utilize reuse. Among others, the **reuse efficiency** was defined. It was derived from the model the return of investment from the reuse economics model. It was presented as:

$$N(C_{NR} - C_R)/C_D \quad (2.2)$$

where

- N denotes the number of systems, version, or products developed with the reusable assets (usage count)
- C_{NR} denotes the cost of developing new assets without reuse
- C_R denotes the cost of using (finding, evaluating, adapting, etc.) reusable assets
- C_D denotes the cost of domain engineering (acquiring or developing assets for reuse, building a reuse infrastructure)

Although this measurement puts together the desired variables and provides a suitable comparative measure, its objective is to evaluate the

degree of reuse for an entire set of assets. It handles the cost of producing the reusable asset differently than it is desired for our purpose. The relationship should be more focused on the reuse efficiency of an individual asset.

Therefore, given the basic formula defined in Equation 2.1, by substitution for actual variables from Equation 2.2 in a way that:

- *productivity cost using reusable asset* $\sim N * C_{NR}$
- *asset production cost* $\sim C_D$
- *productivity cost without reuse* $\sim N * C_R$

and some manipulation, we get a modified formula for reuse efficiency η :

$$\eta = \frac{NC_{NR}}{NC_R + C_D} \quad (2.3)$$

The productivity cost grows with increasing number of repeats. The Figure 2.7 schematically illustrates the dependency of productivity costs on number of implementations. Thus, if $\eta > 1$ (for given N), then it is more efficient to create the reusable asset instead of implementing the system part without reuse. From this point (*pay-off threshold*, N_1), the investment in producing the asset has paid off, and subsequent uses actually save costs.

The graph shows that both approaches have some initial costs. That results from the need of some time to learn the requirements, options and constraints given by the technology and domain. In fact, the behaviour of the functions is not strictly linear because the efficiency usually improves with subsequent implementation iterations.

2.3.2 Measuring Costs

The reuse efficiency formula proposed in the previous section contains costs in terms of an effort to create a specific part of the system or functionality. However, it was not indicated how such costs are derived. In the case of this thesis, a way to estimate the costs of development solutions in BPM needs to be found.

Jørgensen and Shepperd conducted a rather comprehensive review [39] of software development cost estimation approaches that have been dealt with in past years. Among others, they recognized following categories:

Regression: mostly algorithmic cost models based on a regression formula derived from historical project data, e.g. COCOMO [12]

2. REUSABILITY IN BPM

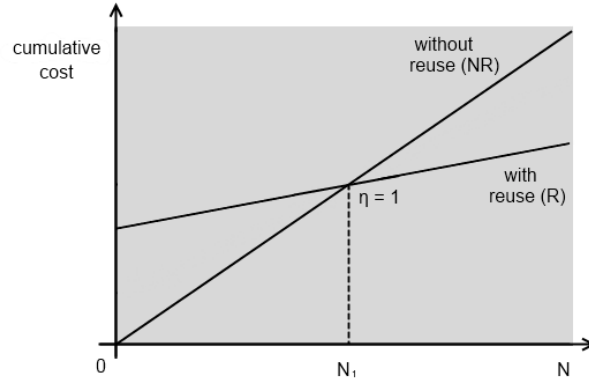


Figure 2.7: Cumulative costs comparison. R denotes reusable asset, NR denotes non-reusable implementation, N denotes number of reuses, $\eta = 1$ is the point where the reuse becomes pay off

Analogy: approaches based on evaluating analogies in other methods

Expert judgement: based on strategies for evaluation made by specialists

Work breakdown: based on decomposition of activities

Function point: based on estimations of functions, use cases, or features

As it is described in the following chapter, the development of BPM applications has its special characteristics. From the listed categories, the one that fits the most appears to be work breakdown structure-based methods. The other approaches would be somewhat difficult to apply for various reasons; there is, for example, the dependency on counting source lines of code which is not applicable because BPM suites tend to provide graphical-based IDE. Similarly, function point based evaluation does not conform to specifics of the BPM related development.

The proposed method to estimate the cost is based on work breakdown structure (WBS). It is a commonly used technique (described in [35]) for subdividing the effort required to achieve an objective. In WBS, the work is divided into fine-grained activities that permit to realistically estimate the cost to finish them.

This works as a top-down approach. That means that it starts with a 100% portion of the entire statement of work and then it follows by splitting that up into the parts until we have a set of assessable work tasks. In our case, WBS is used in a "reversed mode", so that we start at the lowest level and build up the final deliverable by a composition of the tasks. Therefore,

the total cost for achieving the original goal is not represented as 100% of planned outcome but as a sum of its components' costs.

The estimation of the lowest level activities should be done with respect to cost of other activities within the same project. For this purpose, I defined a unit of work (UOW) that represents a comparable amount of work. Thus, if one activity is estimated for 1 UOW and second activity needs 4 UOWs, then the second activity is four times more demanding than the first one. The UOW does not necessarily represent the amount of time needed to finish an activity by a person. However, if appropriate conditions are properly specified, a coefficient can be determined to enable to translate the work units into time required to perform a task.

Vaníček in his book [64] stated that in Stevens' typology of measurement scales, the most of the quantitative scales used for measuring the costs of software development correspond to the *ratio scale* type. This ratio scale allows to compose, compare and also multiply the units of this scale. These findings support the measurement method I derived for measuring costs of development in BPM. Because the proposed measurement is similar to counting the lines of code (they are just substituted for another entity), it is in accordance with the ratio scale type.

2.3.3 Comparison to Empirical Data

Jacobson et al. in [36] analysed reuse programs in various software companies and combined them with their numerous experiences to distil several rules that may serve as guidelines about reuse (expression derived from the model in the previous section that corresponds to the statement is in the brackets):

1. *A component has to be used three to fives times in application projects to recover the initial cost of creating it and the ongoing cost of supporting it. ($N_1 = \langle 3, 5 \rangle$)*
2. *It costs 1.5 to 3 times as much to create and support a reusable component as it does to implement a similar component for a single application. ($C_D/C_{NR} = \langle 1.5, 3 \rangle$)*
3. *It costs only one-quarter as much to use a reusable component as to develop a new one from scratch. ($C_R/C_{NR} = 0.25$)*

Moreover, they stated that *"It takes two or three product cycles—usually about three years—before the benefits of reuse become significant."* In this thesis, I evaluate reuse in our projects, and compare the results to the experience of Jacobson et al.

2.4 Summary

The motivation for leveraging software reuse in software engineering may resemble the reasons for adopting Business Process Management in organization. Through its methods there can be achieved better quality of delivered outcomes in less time while making less effort. The software reuse practices have various characteristics determining their applicability for different purposes and conditions.

One of the well-established approaches to software reuse, Asset-Based Development, may serve as a convenient methodology basis for introducing reuse practices into BPM program implementations in order to improve agility, cost-effectiveness and quality assurance.

As a way for evaluating reuse efficiency, I have designed a simple metric. It is based on calculation of costs using decomposition of implementation procedure into quantifiable activities.

Assessment of Reuse in Selected BPM Suites

The goal of this chapter is to select of BPMS product that would enable the development of executable business processes utilizing principles of software reuse. A particular focus is put on the IBM Business Process Manager which stands among the most advanced currently offered BPMS products. A general overview of the product is presented along with a description of the development style and capabilities facilitating reuse. At the end of the chapter, a brief comparison of two other BPMS solutions which represent different approaches to the BPM application development is made.

3.1 Introduction

The selection of the right BPMS technology to adopt in an organization is generally a very complex task. Currently, despite a wave of mergers that happened at the BPMS market, there still exist over 70 vendors offering various sorts of BPMS solutions. Each product is targeting the customers with its own strengths which makes the customers think about their key requirements on the BPMS thoroughly. [57]

The assessment procedure of such an important part of the IT infrastructure requires a well-considered strategy because it has a significant impact on the entire organization. It can be performed in different ways, spanning from relying on intuition of responsible individuals through reviewing surveys, comparisons and market overviews carried out by independent research companies like Forrester, Gartner, and IDC, to using some comprehensive selection framework (e.g. [44], [11]). The final choice of the

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

most suitable BPMS technology is partially predetermined by the conditions given by the customer's requirements and constraints given by the business and the technological environment. Since the full-scale comparison of the BPMS products is not within the scope of this thesis, only a brief overview (except for IBM BPM) of the preselected BPMS products is presented. The evaluation was done with focus on the reuse capabilities.

3.2 IBM Business Process Manager

IBM Business Process Manager (IBM BPM) is a business process management suite developed and marketed by International Business Machines (IBM). It consists of a complete set of tools supporting the entire business process life cycle - process modelling, design, execution, analysis, and optimization.

IBM BPM was not originally developed by IBM. It was all started by a company named Lombardi Software which created a quite progressively developed BPM suite called TeamWorks. Lombardi Software was acquired by IBM in 2010, and the BPMS was renamed to WebSphere Lombardi Edition. Since then, the product was incorporated into IBM's product portfolio under the WebSphere brand, and it was merged with IBM's former top BPM product WebSphere Process Server to create IBM BPM. The latest version 8.0.1 was released in November 2012. The evolution of the product is depicted in Figure 3.1. According to Forrester, IBM Business Process Manager stands as a BPMS market leader delivering *"a unified experience for building enterprise-scale programs"*[57] (see Figure 3.2).

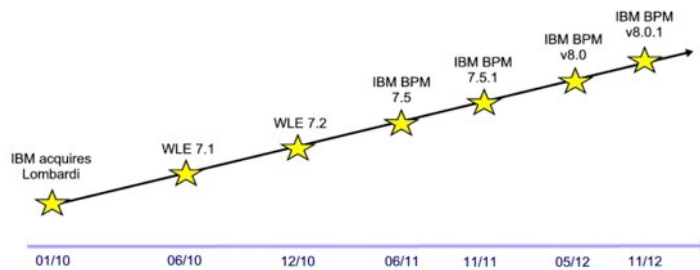


Figure 3.1: Evolution of IBM Business Process Manager [43]

The conception of IBM BPM is to bring the customer the best possible value in terms of improving efficiency and visibility of the organization's processes along with a higher adaptability to the changes. This is

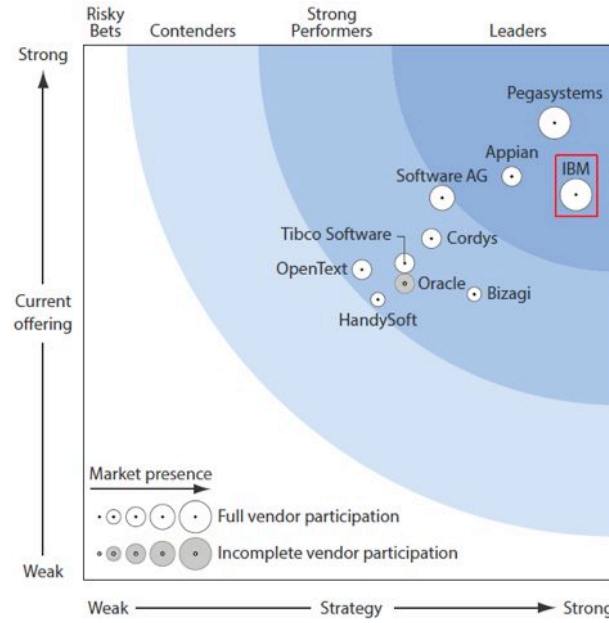


Figure 3.2: Position of IBM Business Process Manager in Forrester Wave - BPM Suites, Q1 2013 [57]

in accord with the latest version of the BPMN (v2.0; see section 1.3.4.1) which establishes the execution aspect of the specification. Thus, only one shared articulation of a model exists without the need of translation to an executable form. This eliminates the round-trip problems seen in other environments. Moreover, the shared model may encourage discussions over a process between business and IT and improve their alignment.

The platform is offered in three different configurations, the selection depends on the phase in a company's BPM maturity life cycle. The Express version is an entry-level product meant for the first BPM project. Although it does contain the complete set of tools, it does not enable scaling the production environment. The Standard version is suitable for full-scale BPM applications without large requirements on process orchestration and extended support for high-volume service operations. For this purpose, the Advanced version which contains Enterprise Service Bus and other means to support enterprise-wide SOA integrations is designed. In this thesis, whenever referring to IBM BPM, the Standard version is always meant.

IBM BPM offers a solution that enables the customer to start with the BPM initiatives in the shortest possible time. This is supported by providing plenty of out-of-the-box features that are well integrated. The sources used for composing this section comprised of IBM BPM documentation

[29], excellent book of Neil Kolban about IBM BPM [43], as well as my own practical experience gained during several BPM projects.

3.2.1 Architectural overview

IBM BPM consists of a collection of several components. They include a unified BPM repository, tools for authors, administrators, and users, and a runtime platform. Each component serves a distinct purpose and they are employed at different phases of a BPM life cycle. The overview of the architecture is shown in Figure 3.3.

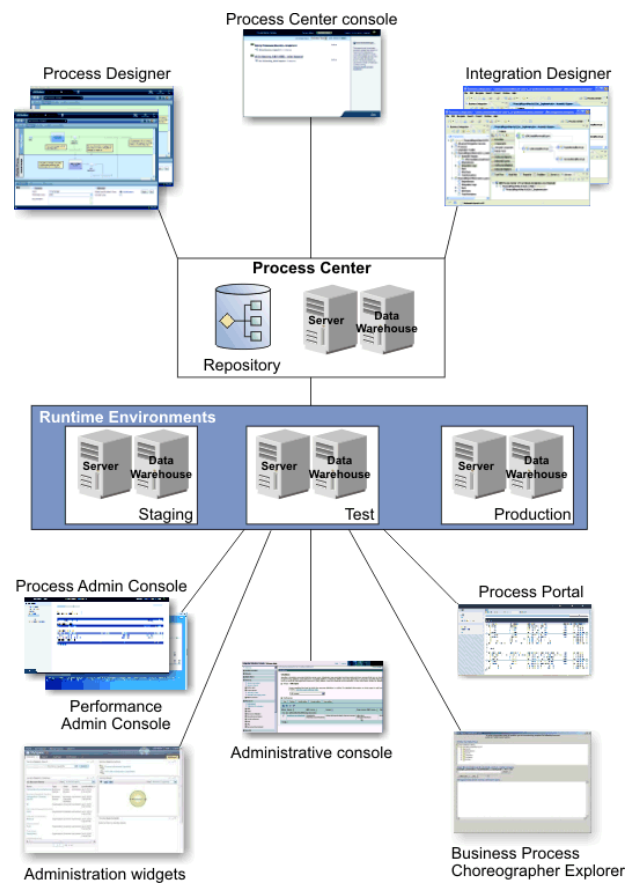


Figure 3.3: IBM Business Process Manager - Architecture Overview [29]

3.2.1.1 Process Applications

The cornerstone of IBM BPM is a *Process Application (PA)*. It can be seen as a project or a container for a solution. It has a name, identifier, tags,

and description and it contains entire set of artifacts that form the actual process model design and implementation. The authoring environment for Process Applications is called the Process Designer. Process Applications are stored within a central repository component called the Process Center. When a PA is ready, it can be deployed to the runtime environment, the Process Server. Users can access their tasks in the Process Portal, and administrators use the Process Administration Console for managing the operations of the process instances.

At this point, the components of IBM BPM should be described in more detail:

Process Server

Process Servers (PS) are engines which run the processes authored in Process Developer and stored in Process Center. Each environment (like a test or production) has its own Process Server with its own deployed processes enabled to be executed. The Process Center also includes a Process Server for the execution of the developed Process Applications. Process Server is implemented by IBM WebSphere Application Server (WAS). The IBM BPM run-time is Java and engineered utilizing Java EE framework.

Process Center

Process Center (PC) includes a repository for all processes, services, and other artifacts created in Process Designer. The Process Center allows to create the assets in a cooperative manner. That is, there always exists always only one copy of a definition and developers and process analysts can edit the same Process Application at the same time. No code or data is copied or stored at client's side, everything is submitted to the server immediately after pushing the Save button. Using this approach has significant consequences. Because the modelling, development and monitoring share the same process representation (see Figure 3.4), there is no need for reflection of changes after each edit of the model. That eliminates a major source of mistakes and misalignments between business and developers. Also, unlike in common version control systems where the check-out/check-in procedure may lead to necessary merges of the code, such conflicts cannot emerge here.

The artifacts within the Process Center are stored in tables within a standard relational database. This database is created and configured during the product installation. The Process Center may be accessed through Process Designer or web interface, as it is shown in Figure

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

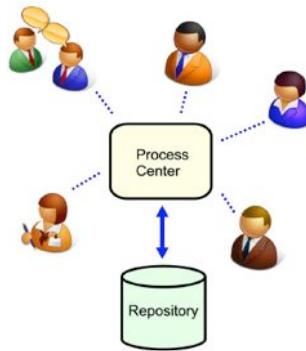


Figure 3.4: IBM Process Center shared repository [43]

3.5. The product allows almost effortless export and import of the entire Process Applications, so they can be easily transferred between the two Process Centers.

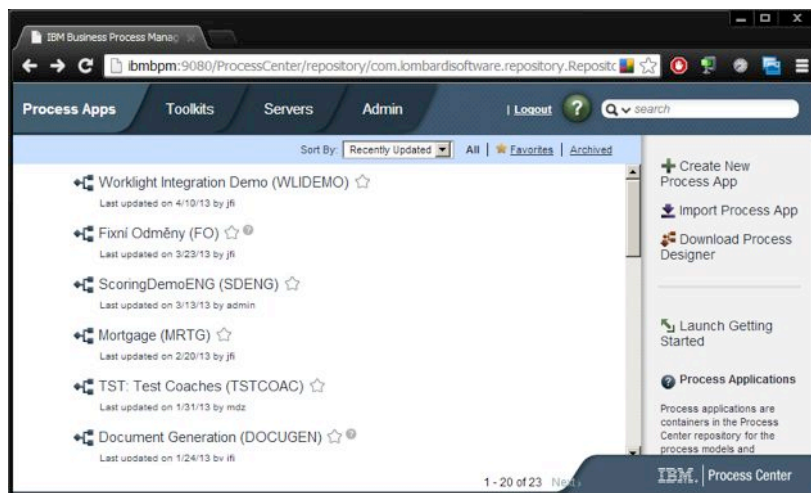


Figure 3.5: IBM Process Center main view

Process Designer

Process Designer (PD) is an integrated development environment designed for modelling, design, implementation, debugging, and testing Process Applications. It is an easy-to-use graphics-oriented tool used to perform sequences actions of which a business process is composed. Multiple Process Designers connect to a single instance of a Process Center where they access the repository of Process Applications. There is no exposed ability to change which Process Center instance

the Process Designer should communicate. This is by design as the intent is to have always only one Process Center in the environment. Similarly to other IDEs, Process Designer interface enables different work modes supported by views that can be switched any time. These views show the process artifacts from different perspectives:

Designer - design and construction of a solution (typically performed by an analyst followed by a developer)

Inspector - debugging and monitoring of a solution (used by a developer for unit and end-to-end testing)

Optimizer - examining process performance testing through analysis, simulation and optimization (performed by an analyst)

Single work environment used throughout the process life cycle supports the idea of a shared model. Process Designer is shown in Figure 3.6.

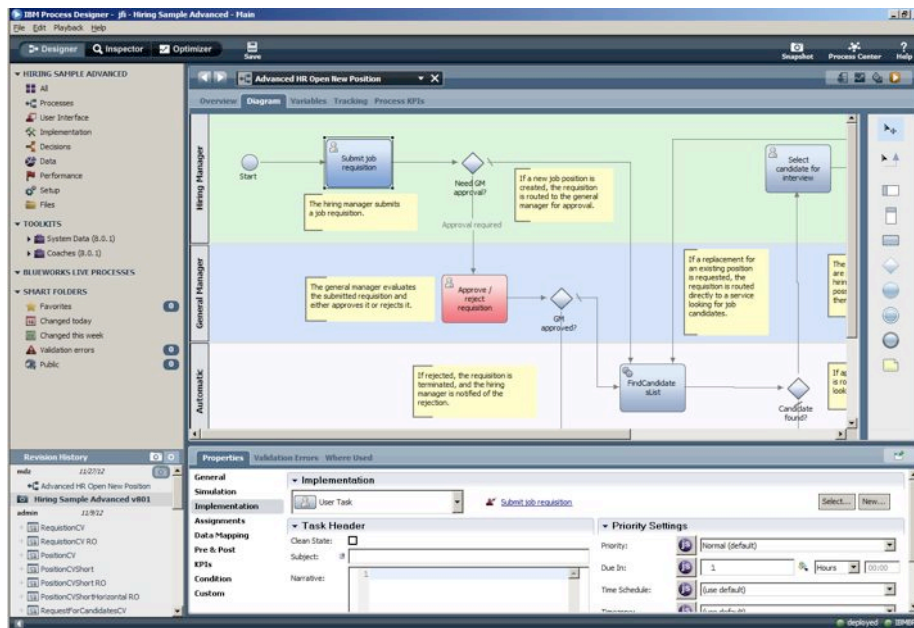


Figure 3.6: IBM Process Designer

Performance Data Warehouse

The Performance Data Warehouse is a database responsible for collecting and managing data originated by Process Server instances. This data is gathered throughout the execution of the process instances and it forms the main source for the reporting of the outcome

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

of processes. Since the queries used for reporting are computationally demanding, the Performance Data Warehouse was introduced to separate the historical data from the process definitions metadata.

Process Portal

Process Portal (or Portal) is a web component that serves the end users as an entry point for their interaction with the system. Primarily, the process participant can view and manage his task list – the list of work items assigned to the him/her that await his/her action. Together with the task there are some more information, such as the due date, priority of the task and visualization of the process model to enable the user a better orientation in the process. There is also a possibility to start a new process instance. The task completion is done through rich web forms (called the Coaches) designed by the process developer. The whole range of performance dashboards contains real-time reporting data of the process instances, workloads, or KPIs that relate to the user, where his/her team or the specific processes can be viewed. Moreover, since version 8.0, there were introduced new social capabilities. For example, to each process task there can be assigned a subject-matter expert who can provide a real-time guidance with the completion of the task.

Blueworks Live

Blueworks Live is a web-based discovery and modelling tool ([27]) that enables involvement of end users, managers and subject-matter experts in the early phases of BPM projects like discovery and modelling without the need of installation and configuration of an IDE. It is provided as Software As A Service (SaaS) application accessible from the web. The intention of Blueworks Live is to make the usage very intuitive in order to encourage the end users and managers to collaborate with analysts in the early stages of the BPM life cycle. The mapping is done just by putting down all the knowledge originally spread across the entire organization. The application enables to use the pre-built templates of processes, and to export the project to wide range of commonly used formats. To fasten the development process, the process developer/analyst may subscribe the Process Designer to the Blueworks Live account and directly access the captured high-level business process models.

3.2.1.2 Data Sources in IBM BPM

Within the process, there are usually three main types of the data. First, the process definition metadata, stored and managed within the process repository. Second, the historical data of process instances used for reporting purposes (stored in a Performance Data Warehouse). And third, the business data related to the process, in IBM BPM often called System of Record (SOR)¹. The SOR is implemented by a separate database usually created for the purpose of storing the business data in accordance to the "Single Source Of Truth" paradigm². There is often a requirement to access the business data from other systems besides IBM BPM, therefore it is common to wrap the database operations in web services, and thus expose the access to the data to other systems (e.g. via ESB) in the enterprise infrastructure.

3.2.1.3 Version Control

IBM BPM provides sophisticated yet simple to use version control system. It promotes a concept of snapshots which is basically a copy of the state of all the artifacts in the Process Application at the point in time when the snapshot was made. Creation of a snapshot enables to revert back in time to the state of the snapshot. The reversion can be done by a single click, and the Process Application is immediately available for running as it was in the previous state.

A Process Application consists of different types of artifacts that form the entire implementation of the process, including BPDs, adapters for integration, user interface definitions, web services, and so on. All the artifacts are shown in Figure 3.7. Each of these items may be reused.

3.2.2 Process Application Development in IBM BPM

In IBM BPM, business process modelling is realized in a specific manner. The goal is to hide the code from the business analyst or the cooperating business participant to keep them focused on the most important task - as accurate capturing of the process model as possible. Therefore the complex source code remains hidden from the user as far as he stays on the process model level. However, the business process analyst should be able to handle

¹For more about System-of-Record architecture, see [54]

²It is the practice of structuring information models in a way that every data element is stored exactly once and linking is done only by reference.

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES



Figure 3.7: IBM Process Designer artifacts

basic implementation steps, especially those that may incur changes of the requirements (such as building user forms).

3.2.2.1 Business Process Definition

The process model is called *Business Process Definition (BPD)* and it is the core artifact of the IBM BPM. The model is represented as BPMN ¹ process diagram (see section 1.3.4.1 for more information) which provides a complete overview of the process flow. The model is created by simple drag-and-drop operations that place the diagram elements on the canvas and link the elements to the desired process flow. Each element can be configured in a way that corresponds to its type. For example, for an activity, the following items can be specified:

- *Type of task* - User Task, System Task, Decision Task, Subprocess, Server Script, etc.
- *Service that implements the task* - by selecting from the implemented artifacts
- *Assignment* - to which role or user the task is routed (applies to User Tasks)

¹v2.0; Although IBM BPM does not conform to the BPMN specification in its entirety, it provides enough modelling capabilities to represent all the flow patterns that may occur.

- *Data Mapping* - of input and output variables

An example of a BPD is shown in Figure 3.6.

3.2.2.2 Business Objects

Each business process needs to work with *business data*. In IBM BPM, it is possible to define variable types called Business Objects (BO) that are shared across the Process Application. These Business Objects may be simple (primitive types provided by the product) or complex (structured types defined by the developer). The complex Business Objects may be composed into hierarchical structures that comprise of other simple and complex Business Objects, which can also form lists. IBM BPM does not support the inheritance or polymorphism of Business Objects. The variable scope is always limited to the artifact in which the variable is defined (BPD, Service, Server script). Figure 3.8 shows definition of a Person Business Object.

The screenshot shows the 'Person' Business Object definition window. It is divided into four main sections:

- Common:**
 - Name: Person
 - Modified: fibicjar (Jan 8, 2013 11:54:03 PM)
 - Documentation: Click [Edit](#) to add or edit text. (Edit)
- Behavior:**
 - Definition Type: Complex Structure Type
 - Shared Object: ☐
- Parameters:**
 - A list of parameters with expand/collapse icons: type (PersonType), id (String), login (String), title (String), titlesBefore (String), firstName (String), lastName (String), titlesAfter (String), email (String), employee (Employee), external (ExternalPerson), affiliation (String), isRepresentative (Boolean), cooperations (Person) (List), and student (Student).
 - Buttons: Add, Remove, Up, Down.
- Parameter Properties:**
 - Name: type
 - Is List: ☐
 - Variable Type: PersonType (with Select... and New... buttons)
 - Documentation: Click [Edit](#) to add or edit text. (Edit)

Figure 3.8: Definition of Person Business Object

3.2.2.3 User Interfaces

Since the release of IBM BPM version 8.0.1, the product gained new capabilities that enhanced reuse options of the *front-end* development. Implementing the user interfaces is based on composing elements implemented with web technologies like HTML, CSS, JavaScript, Dojo Toolkit framework[21] (or other JavaScript framework) and underlying business data structures. Thus built modern front-ends utilizing asynchronous service calls provide a very decent user experience and efficiency of fulfilling human tasks.

The interaction with the user is performed through forms designed in the Process Designer. Forms (in IBM BPM called **Coaches**) can only be included in a specific artifact type, the Human Service. When editing a Human Service, the IDE is extended by an additional Coach tab specialized for Coach design. The Coaches are designed in a way that is very similar to designing the BPDs, or services – drag-and-dropping visual (or even strictly functional) building blocks from the palette to the canvas area.

The building blocks are called the **Coach Views (CV)**. The IBM BPM is delivered with a set of stock Coach Views controls that provide basic functionality for creating Coaches. Among others, these stock controls are: (input) Text field, Select (drop-down list), Button, Check box, Data Time Picker, Radio Button, Tabs, Text Area, etc. The Coach View is defined by four perspectives:

Overview - description of the artifact, visual appearance in the design or special properties. This section is especially important if the Coach View is intended for reuse.

Behaviour - scripts (HTML, CSS, JavaScript, Dojo and other code) defining the behaviour of the Coach View; it enables to implement dynamic behaviour of the user interface using AJAX calls and event-based communication on the client-side

Variables - business data binding variables and variables that define configuration exposed outside the Coach View

Layout - positioning and configuration of the elements used by the Coach View

Coach Views that visualize business data (e.g. Text field) or use them as an input, needs to be bound to variables of a specified type. Therefore, the type of the bound variable defines the interface for using the Coach View. Also, the exposed configuration of a Coach View must specify the data (or service) types. For example, the stock Coach View called *Select*

enables through its configuration to choose whether a user may select single or multiple items, and a service which provides the list of items available for the selection.

The concept of Coach Views enables to form hierarchical structures by nesting them, using a special container called Content Box (similarly to nesting `<div>` tags in HTML). Thus, it is possible to create an entire hierarchy of components for creating the user interfaces which are reusable across multiple projects if defined conveniently.

For example, a Car Insurance Claim form may consist of components (Coach Views) like Personal Details, Car Details, Insurance Details, Incident Description, etc. Each of these Coach Views may be composed of other Coach Views. For example, Personal Details Coach View may contain simple Coach Views like First Name (text box), or Date of Birth (date picker), as well as Address which is a Coach View comprising of other Coach Views. Once the Coach Views are defined, they are reusable in other Coaches or Coach Views in the Process Application.

3.2.3 Capabilities Supporting Reuse

In accordance to the goals of this thesis I examined the product for the capabilities that support the development of reusable assets and reusability principles.

3.2.3.1 Toolkits

One of the major principles used in IBM BPM is using a unified repository for the assets. This repository (Process Center) contains a set of Process Applications that each represent a container. Process Applications contain artifacts that together form a logical unit implementing a business process or an entire set of business processes. The artifacts defined within a Process Application cannot be used by another Process Application. For the purpose of reuse, the Process Center enables to create Toolkits. Unlike a Process Application, a Toolkit does not result in a deployable application. Instead, the contents of the Toolkit can be used by one or more Process Applications.

The product contains several built-in Toolkits, that offer basic functionality provided out-of-the-box. These are:

System Data Toolkit contains core definitions for data types, default service definitions, basic database and web services integration, etc.

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

System Governance enables building of governance processes to control the installation of new process versions and so on.

Coaches contains stock Coach Views ready to use in Coaches.

Content Management provides access to Enterprise Content Management types and services.

The concept of Toolkit act as reusable asset package in the business process application development. Any Toolkit stored in the Process Center can be easily imported to the Process Application which makes all the included artifacts immediately available for use. Just like Process Applications, Toolkits are versioned using snapshots. When the dependency on the Toolkit is created, the specific snapshot of the Toolkit needs to be determined. Adding and changing the version of dependency procedure is very simple; it can be done by a single click in the Process Designer (see 3.9).

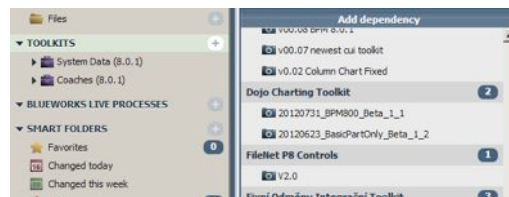


Figure 3.9: IBM Process Designer - Adding a dependency

The Toolkits can also be used in other Toolkits. That makes it possible to create entire Toolkit dependency trees. (see example in Figure)The Process Application may even have dependencies on more versions of the same Toolkit. This does not cause any interferences or conflicts as two separate Toolkit versions act like two different Toolkits.

Although the typical IBM BPM environment contains only one Process Center, it might be advantageous to share the Toolkits among different instances of the Process Center. The Provider PC exposes the Toolkit available for other PCs. The Consumer PC subscribes the shared Toolkit to enable importing a copy to its own repository. The Consumer PC is notified whenever a new version of the Toolkit is released by the Provider PC. In this way, the development of the reusable assets may be dispersed among separate teams collaborating on large BPM projects. For example, one team may focus on the user interfaces while the other one focuses on the integration adapters. The topology may look like the one in Figure 3.11.

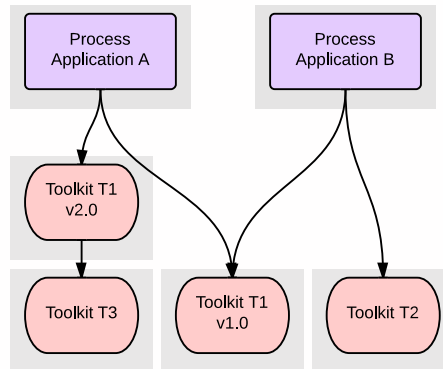


Figure 3.10: Toolkit dependencies

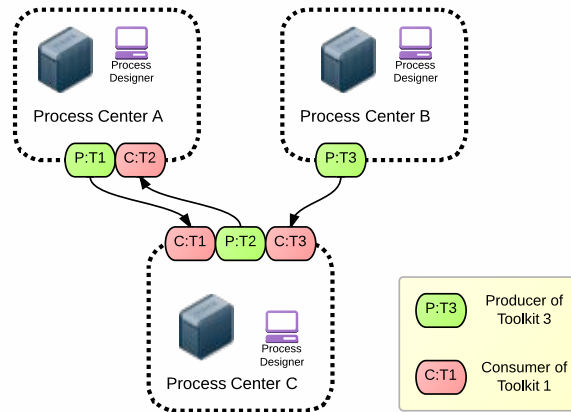


Figure 3.11: Toolkit sharing among dispersed Process Centers

3.2.3.2 Working with Assets

To facilitate the reuse in the process development, the product offers various ways to utilize the shared repository assets. The reusable asset in IBM BPM life cycle undergoes a life cycle similar to the one described in section 2.2.1. Most of the use cases can be applied using Process Center and Process Designer components. The mapping of the use cases to the components and their functions shows Table 3.1.

In IBM BPM, the asset package has a form of a Toolkit. As a reusable asset we mean the artifacts that are placed in the Toolkit. Each such artifact can be used separately by the asset Consumer, however, the Toolkit must always be imported whole. Each artifact includes a description to provide

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

the reference to the potential Consumer. For this purpose, there exist three main artifact metadata:

Name - Since the Process Designer provides excellent type-ahead suggestions based on the artifact name, a properly chosen name may greatly facilitate the asset consumption. The development team should establish naming conventions so the name of the desired asset could be easily deducted.

Tags - Tags provide the capability to classify the artifacts by marking them with a keyword from a defined set. IBM BPM provides an elegant solution for searching and sorting the artifacts with filters based on these Tags. The tag may denote the development status of the artifact, as well as its context, required resources, used technology, or class designed purpose.

Description - The description should contain information essential for potential Consumer of the artifact. These may be usage instructions, list of related assets, changelog, constraints, etc. It may also contain links to external repositories or web pages.

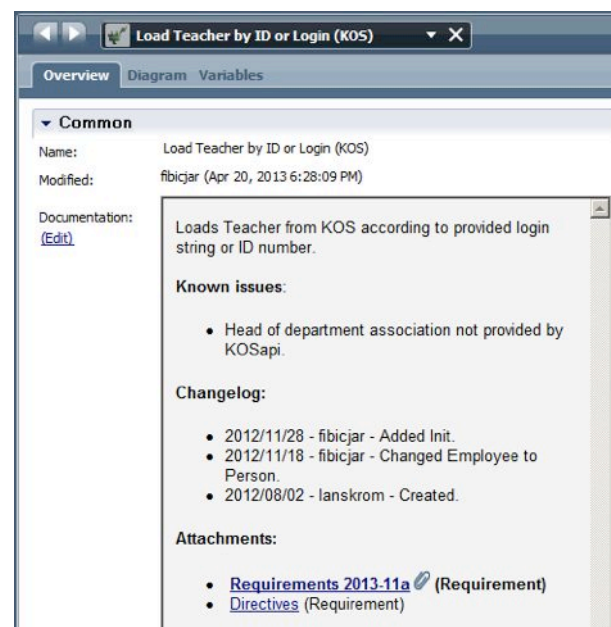


Figure 3.12: Process Application artifact metadata

The artifact description is shown in Figure 3.12.

All three metadata fields are searchable by the Process Center search engine. The search criteria is entered by a single search string. A simple syntax can be utilized to specify the type of the searched artifact along with logical operators. The engine searches through the entire repository, including the subscribed Toolkits from external Process Centers.

Moreover, in Process Designer, the developer can define dynamic collections called *Smart Folders* which automatically change their content based on the given criteria (tags, last modification date, name pattern, etc.).

Table 3.1: Mapping asset use cases to IBM BPM components and features

Use case	Component	Supporting feature
Harvest	Process Designer	<ul style="list-style-type: none"> – Copying items between Process Applications and Toolkits – Define input/output variables
Package	Process Center	<ul style="list-style-type: none"> – Create Toolkit
Catalog	Process Center /Designer	<ul style="list-style-type: none"> – Fill description, assign tags
Purchase /Deliver	Process Designer	<ul style="list-style-type: none"> – Add dependency to a Toolkit
Search	Process Center	<ul style="list-style-type: none"> – Search with filters
Analyze	Process Designer	<ul style="list-style-type: none"> – Read descriptions – View the implementation of the artifacts
Apply	Process Designer	<ul style="list-style-type: none"> – Use (place or assign), map inputs/outputs, specify the real parameter values – Copy & Paste, Incorporate

3.2.3.3 Integration with External Systems

Although IBM BPM provides some built-in adapters for integration with the underlying and cooperating systems, it is necessary to provide a way to utilize external custom-made **software libraries**. Since the entire product is built on the Java platform, it enables integration with other custom written Java libraries. The integration is available through invocation of the methods of Java classes. The Java library intended for using needs to be

imported into the Process Application or Toolkit. After this is done, the desired Java class is selected and the Java Integration component automatically discovers all the public methods within that class, including its argument types and returned variable type.

In similar way, it is possible to work with **web services**. The Web Service Integration component is provided with a WSDL URI and the available web services can be automatically discovered. The service is selected and input/output parameters mapped to the process variables. IBM BPM lacks a simple graphical way to map the web service variables that are included in some other BPM suites. However, the mapping can be done by performing XSLT transformation instead. The Web Services can be also very easily created to provide the functions implemented by IBM BPM Services to other systems.

Another way how an external system may communicate with the deployed Process Application instance is the rather comprehensive REST interface supplied by the product. The APIs include services for working with process instances, tasks, models, searching or documentation. The functionality and technical details can be inspected and tested with a supplied BPM REST testing application.

All of the described features enhance the reusability of the system both from the outside and also from the inside. During the BPM implementation practice it was discovered that sometimes the internal server JavaScript API does not provide a certain function, so the REST API needs to be used instead.

3.2.4 IBM BPM Methodology

As it was mentioned in the first chapter, BPM should not be considered only just as a kind of specialized piece of software. The overall impact of Business Process Management initiatives on the entire organization is so significant that BPM should be taken into account by individuals on all managerial levels (strategic, tactical, operational). It always brings a certain change of perspective to the organization. IBM BPM is a product that facilitates not only the impacts of technology adoption but also the associated organizational and cultural changes. This is done by providing methodology, best practices and guidelines on how to approach such movements. The fundamental guidance can be taken from the book [34].

Building large-scale BPM implementations requires proper strategy assessment because it is always a long-term activity. The adoption of BPM principles, like everything new, needs to be absorbed gradually. The first step is usually undertaking a pilot BPM project. The outcomes of this pro-

ject are crucial since they influence the success of an enterprise-wide BPM program. The first BPM project provides an opportunity to identify the best practices, roles, methodology, and baseline against which the future state can be measured. When the first success is verified, the scope of BPM adoption can grow through several projects to a full-scale BPM program as the BPM maturity is increased (see Figure 3.13).

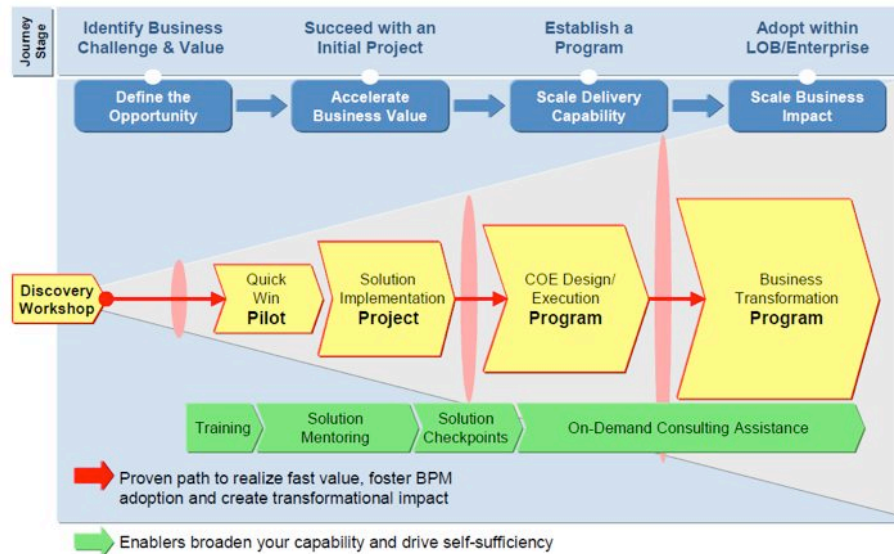


Figure 3.13: Scaling from a BPM project to a program [34]

On the top of the experience gained through BPM projects, it is recommended to find a Center of Excellence (CoE) that would support all aspects of evolving BPM in the organization [32]. One of the tasks of CoE is to develop a system of continuous education. A successful BPM program relies on a wide range of skill set from all parts of the organization, including process participants, process owners, analysts, developers, administrators and architects. Among all the roles, it is the process architect who forms the backbone of successful BPM program [33]. It is the role that links the requirements of the business people to the understanding of technical aspects of IT.

Those who design, develop, implement, administer and use a software system usually tend to collaborate on dealing with the issues that emerge. That is the motivation that leads to creating communities that cooperate to share knowledge about the software product. When a problem arises, an individual striving for a solution has a number of options where to look for help. Besides vendor support and consultancy, it may be a forum, wiki

pages, mailing lists, team members, product manual etc. For IBM BPM, the most valuable resources are developerWorks forum [30], and IBM BPM Community wiki [28].

3.3 Comparison of Reuse Approach in Other BPM Solutions

Both commercial and open-source BPMS vendors have established their own approaches to the phases of the BPM life cycle that they inscribed into the BPMS technology that they produce. On the following representatives of BPMS, one from commercial domain (Bizagi Go) and one from open-source software (Acitiviti BPM Platform), different approaches to the implementation of executable business processes is presented.

3.3.1 Bizagi

Bizagi's BPMS offering is called Bizagi Go and its latest version 10 was released in December 2012 [10]. Although its market presence is not among the top ten BPMS vendors, the solution is valued for its undisputed qualities, especially in its approach to linking business process and external data sources [57]. Their sales model is supported by offering a free, easy to use process modelling tool Bizagi Process Modeler which often encourages the customers to follow up with process automation using Bizagi Go.

3.3.1.1 Key Advantage

The Bizagi BPMS platform enables the process developer to create a virtualized data model that can be synchronized with the source business data and legacy applications using very little effort. The process model does not have to be bloated with numerous technical calls of services providing the data sending, retrieval, or transformations and thus it better separates the business-oriented view from the technical perspective. This way, the reusability of the data services is highly improved because the access is provided by the system itself.

3.3.1.2 Opportunities for Reuse

Bizagi is offered in two editions - Xpress and Enterprise. Among other things, they differ in reuse possibilities within the process development.

3.3. Comparison of Reuse Approach in Other BPM Solutions

Xpress Edition allows to reuse only user interfaces while the reuse of Enterprise Edition capabilities covers some additional Bizagi components.

When we look at the approach of Bizagi for building the user interfaces (which is a common subject of reuse), the user forms are composed from a comprehensive set of UI elements that allow a certain range of configuration, dynamic data hiding and data validation. Certain parts of the forms created this way might be reused (similarly to creating Coach Views that group elements in IBM BPM). In addition to the out-of-the-box form elements, Bizagi provides a conception of Widgets that provide more advanced UI features like integrated charts or Google Maps. To use a Widget in a form, it needs to be downloaded from Bizagi Widget Store (see Figure 3.14) and imported to the form designer.

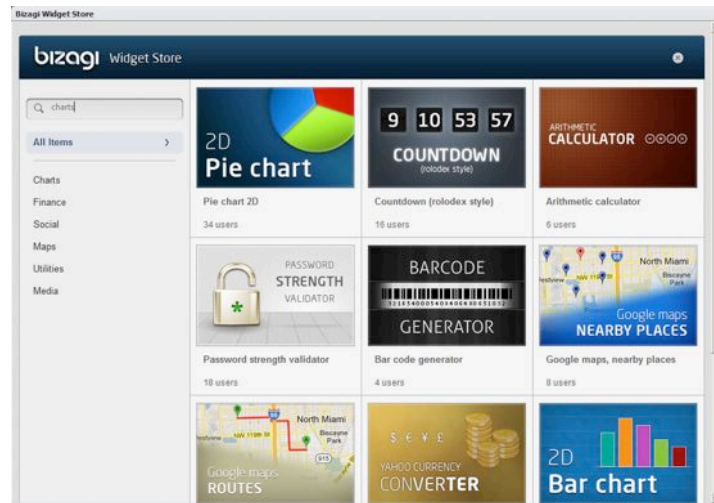


Figure 3.14: Bizagi Widget Store [10]

Although the options for creating user interfaces in Bizagi are quite rich, they do not provide a possibility to create and reuse custom-made components developed using common web technologies like JavaScript frameworks or AJAX.

In addition to user interfaces, Bizagi Enterprise Edition allows to reuse other process artifacts, namely Vocabulary (constant definitions), Business Rules, Expressions or Functions. For interoperability with the external systems, Bizagi provides ways to call Web Services, RESTful services, or custom-coded class libraries. The calls are defined as interfaces reusable on multiple places in the process application. The mapping technique of inputs and outputs to the virtualized data model is convenient and easy to use. [10]

3. ASSESSMENT OF REUSE IN SELECTED BPM SUITES

Although the described possibilities for reuse in Bizagi provide some opportunity for improved implementation efficiency, the product itself does not support packaging of the developed process artifacts and management of such acquired reusable assets. This fact decreases the potential for creation of reuse management programme spanning across multiple projects or business units.

3.3.2 Activiti

Unlike the previously described BPMS products, Activiti BPM Platform [4] is a representative of an open-source software. Open-source BPM suites can act as viable alternative to the commercially bred BPMS products. However, selecting the open-source technology brings the customer different challenges during BPM project implementations.

Activiti is an open-source BPMN 2.0 process engine framework that provides an environment for running business and technical processes. It is a project established by jBPM¹ founder Tom Baeyens and funded by Alfresco². The development started in 2010, the current stable version is 5.12.

A key characteristic of Activiti is that it is developer-focused. The suite consists of several components (see Figure 3.15) that supply functionality supporting the BPM life cycle. The core component is the process engine (Activiti Engine) that implements the BPMN 2.0 specification. It provides comprehensive Java API that is used to implement all the aspects of business processes, for example user task forms, authentication, task management, or repository management. [56]

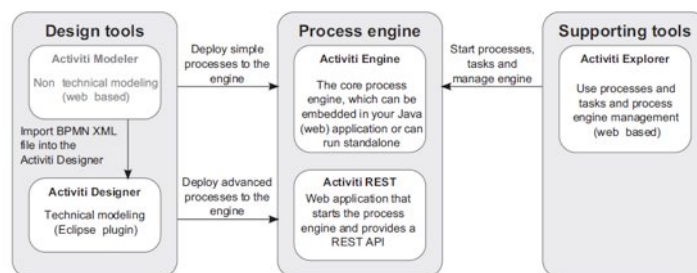


Figure 3.15: Activiti components [56]

Reusability within Activiti corresponds to reuse capabilities of traditional object-oriented programming. Therefore, a vast variety of frame-

¹another open-source BPM platform, <http://www.jboss.org/jbpm/>

²open-source document management system <http://www.alfresco.com/>

works, software libraries, and design patterns may be used to facilitate the development process but it takes some effort to create specific BPM functions from scratch.

Although open-source BPMS solutions have recently made a decent advance in overall BPM product maturity, adopting them might have a considerable drawbacks. One of them is the lack of an out-of-the-box functionality that needs to be custom-made if required.

3.4 Summary and Comparison

Selecting the right BPMS has significant impact on a successful adoption of BPM in the organization. Vendors offer various BPM products that supply different approaches to implementation of business processes and also to improving development efficiency by reuse of process artifacts.

Three BPM suites were reviewed to compare their abilities to reuse what was already created before. All of them claim to focus on collaboration between business people and developers on capturing, automating and running business processes. However, they vary significantly not only in the interpretation of this statement.

In both commercial BPMS products, Bizagi and IBM Business Process Manager, the intention is to enable implementation of substantial parts of the process through graphical IDE, including building of the user interfaces. Additionally, a process can be run immediately, without any compilation or deployment. By contrast, everything in Activiti except for the process model itself is created by coding.

Evidently, there are also differences in approach to reusability. While IBM Business Process Manager provides capabilities allowing the creation of custom UI elements, complex integration services or other process artifacts that may be packaged and reused across multiple BPM projects, Bizagi allows only a limited range of reuse within one process application. Also in this aspect Activiti stands aside; the reuse is enabled in the traditional ways of object-oriented programming.

Part II

Practical Part

Case Study

In this chapter, the identified reusability principles are put into practice using the selected BPMS product.

4.1 BPM Programme at FIT

The case study of this thesis was conducted as a part of the business process management adoption initiatives at the Faculty of Information Technology (FIT), Czech Technical University in Prague (CTU). The BPM adoption is one of the major long-term managerial objective of FIT. This effort is also aligned with general strategy of CTU.

Within the preliminary phases of the first BPM projects at FIT, initial steps were taken to start the BPM initiatives. That included the discovery of key process groups within the faculty, definition of development strategy and assessment of the software platform. In the subsequent phase, the business processes that were identified as the most suitable for implementation were put into development process. The implementation team comprised of students and teachers of the faculty that had applied for participation on the BPM project.

Prior to the process implementation itself, there were several conditions to be met.

- A working team needed to be established.
- The development infrastructure needed to be installed and set up.
- The communication channels used within the team needed to be agreed on and arranged.

- The key process participants needed to be informed about upcoming development activities to be ready to provide clarifications and feedback about the developed process.

4.1.1 Implemented BPM Projects

Implementation of two processes, "Final State Examination" and "Diploma Theses", was at the time of writing this thesis in the testing stage (current status of work-as of spring 2013—is depicted in Figure 4.1). These two processes were the first processes that were decided to be implemented using IBM Business Process Manager. Since this was the initial BPM project that was undertaken at the faculty, a lot of practical experience and knowledge about the platform and the faculty (technical and social) environment was gained.

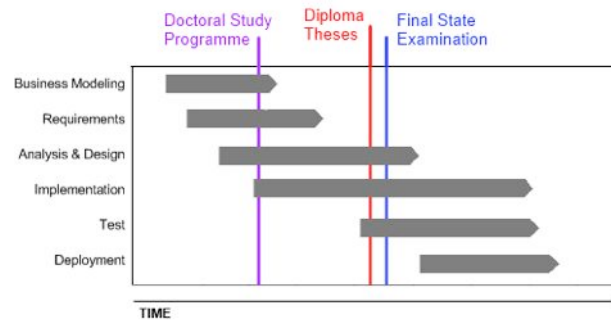


Figure 4.1: BPM project implementation status

From the set of the business processes that had been identified during the process discovery activities at FIT, the processes related to the Doctoral Degree Study Programme (DDSP) were selected for the next implementation phase. These processes formed the whole process group and covered all the activities regarding DDSP, for example admission procedure, student curriculum management, dissertation defence, or comprehensive doctoral examination.

4.1.2 Complications and Support

At the time when this thesis was decided to be written, it was planned by the BPM implementation team that the two initial processes will have been finished by the end of 2012. However, due to several reasons the project schedule was significantly delayed, and because the priority was given to

finishing the initial projects, the work on the DDSP processes had to be postponed by several months. At the time of writing this thesis, the DDSP project was in the initial implementation phase. The situation is shown in Figure 4.1.

The reasons why the projects were delayed are rather typical for initial BPM implementations. That is, the BPM platform is often interoperating with various systems in the infrastructure where it resides. In this case, there were LDAP, databases, ECM system, portal solution (Liferay), legacy system for management study agendas, etc. All these systems needed to be integrated with BPM which consumed significant amount of time of the implementation team. On the other hand, if these issues are handled systematically (sometimes reuse comes in useful), the subsequent projects benefits from the initial work and the implementation progresses a lot faster.

Besides the previously mentioned FIT team that had been established within the faculty, there was one more institution dealing with the BPM on the grounds of CTU that is worth to mentioning. The Center of Knowledge Management¹ at CTU was founded in 2008 to form a Competence Center focused on Business Process Management and related areas applied mainly on academic grounds. Their team of business process analytics and developers possessed an extensive amount of knowledge of BPM. The initial BPM implementation projects were positively influenced by the cooperation established between the teams.

4.2 Approach to Reusability

4.2.1 Asset Harvesting

Throughout the implementation, the teams realized several times that some segments of the implemented processes reoccurred. In most of the cases this happened already during the modelling or design phase because the need for these features was based on certain requirements on the system. However, sometimes the need for reusable component had not been apparent until the developer recognized a repeating patterns during the implementations.

Our general approach to the reusable implementations starts with the investigation to find out whether it is even possible to create a generic component that would be applicable in different parts of the system. If so, the next primary task is to decide if it is worth the effort to implement the type of solution which would bring possibility to simpler reuse of the considered capability. In certain cases, the implementation complexity

¹see <http://www.czm-cvut.cz/>

difference between the reusable and the single-purpose solution might be so significant that implementing the feature from scratch would cost less overall. Sometimes it was possible to determine the level of the asset universality more dynamically – from very generic solutions to not so reusable solutions. When choosing the most suitable solution for the given situation the following factors we considered:

- Complexity of the design and coding tasks required
- Level of reusability of the outcome (e.g. robustness)
- Technical limits (given by platform, programming language, or infrastructure capabilities)
- Level of developer's skills
- Dependencies on other assets

4.2.2 Using Toolkits

The conception of Toolkits has been proven to be a very important feature of the BPMS. Throughout the implementation projects, they us to facilitate reuse of various functionalities and artifacts in more systematic way. Although it was always possible just to make a copy of the desired artifact into the targeting Process Application and make use of it in the same way as at the originating application, every time when it was possible and worthwhile, we moved the artifacts to the external Toolkit. However, the evaluation of efficiency of such decisions were done exclusively based on the developer's intuition, or sometimes after discussions with the rest of the team to verify the benefits of such action.

Table 4.1 shows significant Toolkits that were utilized within the Process Applications. Besides the name of the Toolkit, the table shows its origin, description of the its features, and a number of artifacts it currently contains. This number though also includes items within the Toolkit that are created only for implementing features actually intended to be provided to the consumer. There is no such concept like Java Access Modifiers in IBM BPM, that could be used for hiding the "private" artifacts. However, the possible solution is to apply the "Public" tag on the artifacts intended for consumption.

The Toolkits originated from three main sources - they were either bundled with the product, acquired from some external source (mainly the IBM BPM developers' community) or custom-created. The Toolkit

4.3. Implementation of the Selected Reusable Assets

Toolkit name	Toolkit origin	Purpose	Artifacts count
System Data*	Stock	system provided items	171
CTU Toolkit	Custom	CTU domain-specific items	61
CZM Toolkit	Custom	CZM domain-specific items	12
Content Management	Stock	stock ECM connectors	37
Custom UI	Custom	General-purpose custom-made UI elements	171
PS Toolkit Core	Community	all-purpose utilities (LDAP, database, emails, FTP, filesystem, etc.)	265
KolbanTK	Community	REST call adapter, custom UI elements	170
Alfresco Document Attach	Custom	advanced connector services for Alfresco ECM	41
Change Log	Custom	Process function	10
Jira Integration	Custom	connector to JIRA bug-tracking system	15
Google Integration	Custom	connector to utilize Google Docs API in Coaches	19
FIT LDAP	Custom	domain-specific LDAP connector	4
Legacy Document Attach	Custom	attaching documents to process	15
ORM	Custom	object-relational mapper	17

Table 4.1: Existing Toolkits in CTU Process Center, (*) denotes mandatory Toolkit

delivered with the product provided solid with an expected outcome. They are always thoroughly tested before including into the product and thus their usage was in most cases without negative issues. The community-based Toolkits are usually published "as-is" which sometimes resulted in various issues and thus additional effort to consume such assets. Firstly, the documentation is often insufficient, lacking proper description of functionality or usage. Secondly, the asset may be not tested thoroughly in every possible setting, so it may incur extra debugging to use. And finally, the asset producer may not react on feedback which may in some occasions lead to impossibility to use the asset. By contrast, the Toolkits developed within the team have its creator available to contact personally, so it is easier to consult any issues that emerge.

4.3 Implementation of the Selected Reusable Assets

To evaluate possibilities of reuse in IBM BPM, three assets (that had been created during the BPM projects I took part at CTU) were examined for the degree of reusability which they may deliver. The selection of the particular assets was done in order to show the different approaches and types of the reused assets. The assets I chose for the evaluation are shown in Table 4.2. The table also contains categorization of the assets based on

4. CASE STUDY

subsection 2.1.2.

Name	Substance	Scope	Mode	Technique	Intention	Product
ORM	artifacts	General-purpose	ad-hoc	compositional	Black Box	class library
Person CV	artifacts	Domain-specific	ad-hoc	compositional	Glass Box	GUI component
Change Log	artifacts	General-purpose	ad-hoc	compositional	White Box	GUI component, service pattern

Table 4.2: Assets chosen for evaluation

The table shows that all the presented (and evaluated) assets have the same substance, mode, and technique. That unity in *substance* and *technique* is simply a consequence of the shared technical environment. Thus, the reuse is always done through composition of Toolkits. Although the mode (ad-hoc) is the same for these three assets, there are Toolkits in the repository, that were decided to be created before the opportunity for a use appeared. The *product* aspect of the asset is somewhat self-explanatory but the *scope* and *intention* facets deserve a brief discussion.

The *scope* of the reusable asset has its importance and needs to be considered when developing a Toolkit. There are two main cases:

- **Domain-specific assets.** Such assets are bound to a particular domain for which it is created due to some specific characteristics. In the context of BPM, the domain might be represented by a business unit, a process group, or a set of Process Applications covering related areas. Namely in our case, a faculty was considered as a domain. The Process Applications for this faculty needed specific service connectors that were not needed in other domains/faculties.
- **General-purpose assets.** Assets that are universal in principle; therefore they can be utilized within any domain without limitations for its consumability. An example of such asset package is our Custom UI Toolkit that is invariable to the data it displays.

The significant differences were also in the *intention* asset aspect. Basically, there appeared three ways how to apply the asset:

- **White Box.** It means reuse of components of which internals are changed for the purpose of reuse. White boxes are typically not reused as is, but by adaptation. In process development with IBM BPM, this type of reuse implied copying of the artifact from the Toolkits or other Process Applications and modifying them at the target place. Since in

some cases there was not an option to avoid this step, white box could be at least facilitated by the producer of the asset by preparing of the artifact template, which was included into the Toolkit along with instructions for application. However, this pattern has its considerable drawbacks, it was not practised if there was no other (worthwhile) option.

- **Glass Box.** The term glass-box reuse is used when components are used as-is like black boxes, but their internals can be seen from outside. This is the most common reuse style within IBM BPM, because whenever a Toolkit is imported in the application, all the containing artifacts become accessible for read. The advantage of this approach is that the consumer of the asset may look into the implementation to search for the information of how to use the asset in particular cases when it was neglected in the documentation. Moreover, the consumer may learn from the producer (which happened very often in our team, therefore this was a great feature for skill transfer).
- **Black Box.** Reusing a component as a black box means using it without seeing, knowing or modifying any of its internals. Since the native artifacts of IBM BPM cannot be obfuscated when using the Toolkit, the only case when the reuse was done as black box was when the Toolkit contained some kind of compiled code (e.g. JAR file). In that case, the internal implementation was not accessible to the developer (except for using decompilers).

4.3.1 Object-Relational Mapping

Object-relational mapping (ORM) is in computer science a well-known concept. It is a programming technique that enables conversions of data between different type systems. Typically, it is used for automatic translation of the logical representation of objects used in object-oriented programming into representation based on ER¹ model of relational databases. ORM significantly facilitates persistence of the data used with programming languages.

Although there exists a variety of ORM solutions for commonly used object-oriented languages, IBM BPM lacks such capability for easy to use persistence of its Business Objects. However, in most Process Applications, this type of functionality is required. While the process variables are accessible within the Process Application by a simple data exchange between

¹entity-relationship

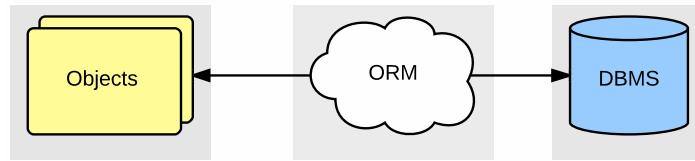


Figure 4.2: Object-relational mapper

the tasks, services, and user interfaces; these business data are not available for any of external systems. Therefore, a dedicated database for the business data is usually created to store the data and to provide access to them for other systems (see more about data sources in IBM BPM in sub-subsection 3.2.1.2). In that case, Process Applications need to utilize some method for accessing an external database.

In the case of IBM BPM, the notion of Business Object (BO) is the object to be mapped to the database. Since the Business Objects form only simple hierarchy of nested objects without abilities like inheritance or polymorphism, the mapping should not be as complex as it is for Java or other object-oriented languages.

IBM BPM offers a built-in capability to connect to the external databases through the underlying WebSphere Application Server (WAS) JDBC definitions (called Data Sources). When a Data Source definition is successfully created on WAS, the database becomes available to use from the processes. In IBM BPM, the database is usually accessed from the Service level. The methods to access the data source are provided by pre-delivered component that is included in the System Data Toolkit. These methods allow to execute single SQL statement, multiple SQL statements in one transaction, or to call stored procedures.

To deal with the described issue, three different solutions are proposed and compared in terms of reusability. The first solution comprises of straightforward use of the connector component included in the product. Building of the SQL statements is done one at a time, so it is rather unflexible for both use and change. The second solution is an improvement of the first one to achieve easier composition of the SQL statements. The third solution is the most advanced and requires a lot more effort to develop. It introduces specialized Java connector for object-relational mapping which runs the database queries by itself and returns the composed Business Object. On the top of this Java connector, a server-side JavaScript methods facilitates the use of the connector with more convenient mapping definitions.

To perform the comparison, a reference database schema and corres-

ponding Business Object system was created. It consists of a **Person** (1) object comprising several attributes of primitive types like **personId**, **name**, or **login**. Besides those, it includes the **Department** object (2) and a list of **Address** objects (3). The **department** among others includes variable **head** (4) (which represents the reference to the Head of Department) of **Person** type. This creates a recursive definition, that may cause problems to the automatic OR mapper. The entire structure can be seen in three different representations - Figure 4.3 shows the Business Objects in IBM BPM, Figure 4.4 shows the UML Class Diagram and Figure 4.5 shows the database schema model diagram.

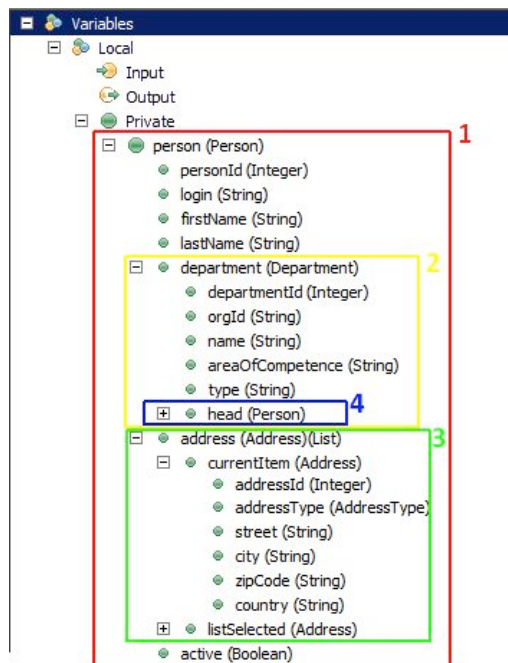


Figure 4.3: **Person** Business Object definition in IBM Process Designer

4.3.1.1 Approach I - Non-reusable Development (from Scratch)

The first solution to the object-relational mapping in IBM BPM uses the built-in database connector without any additional optimizations. The basic implementation of data retrieval service is shown in Figure 4.6. In this implementation, I assume that no reuse is possible, or at least it is not possible to parametrize the service to alter its behaviour. Therefore, in case there is some variation needed to be implemented, the service needs to be created again from scratch.

4. CASE STUDY

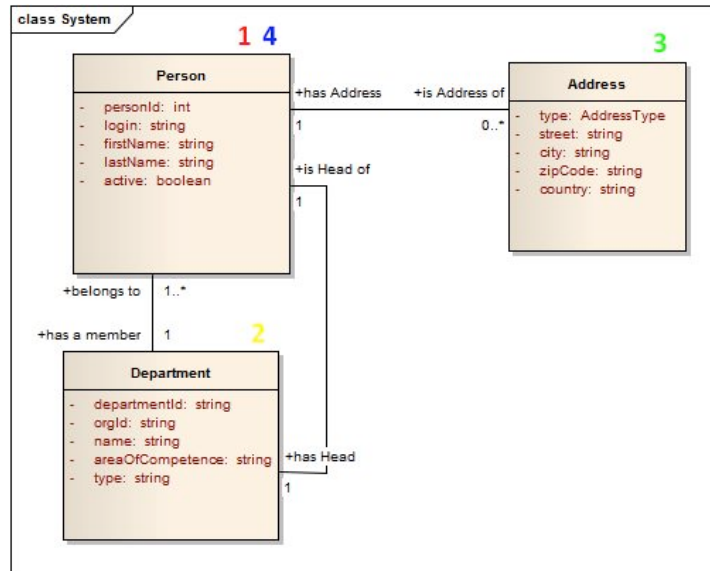


Figure 4.4: Person - UML Class diagram

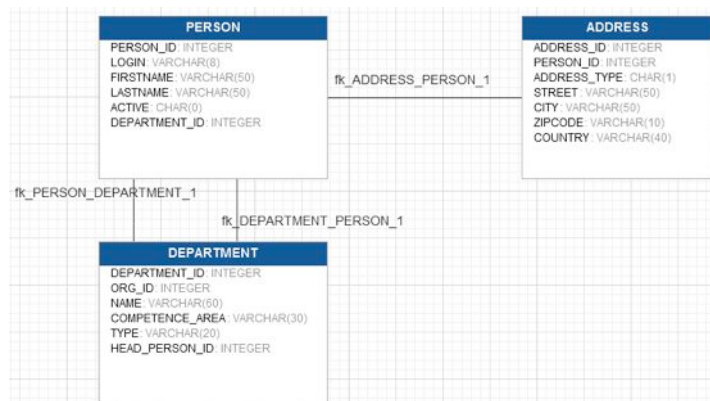


Figure 4.5: Person - database schema model diagram

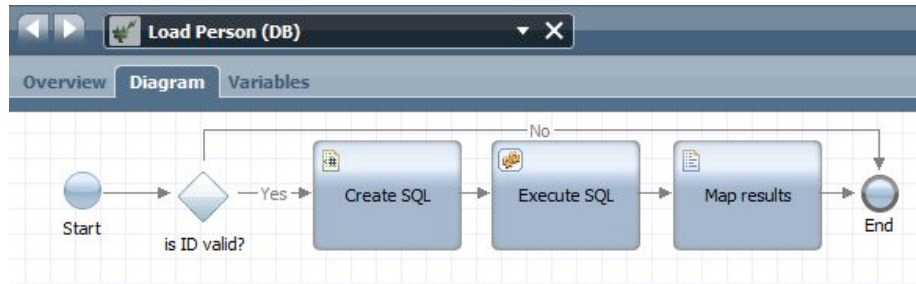


Figure 4.6: Basic DB data retrieval and mapping implementation

Implementation of the `Load Person (DB)` Integration Service is simple. It awaits 1 input parameter `personId`, which is the primary key to table; and returns an object of the type `Person`, if such `personId` exists in the table. The implementation consists of 4 steps:

1. **Is ID valid? (Decision Gateway)**. Check if the input variable was passed to the function. If it is undefined, the service flow continuous straight to the End.
2. **Create SQL (Server Scriptlet)**. Builds the SQL query:


```
SELECT LOGIN, FIRSTNAME, LASTNAME, ACTIVE FROM PERSON WHERE PERSON_ID = <#=#tw.local
.personId #>
```
3. **Execute SQL (Nested Service call)**. Call the `SQL Execute Statement` provided with the created SQL.
4. **Map results (Server Script)**. Since the built-in SQL connector returns the data serialized in XML, the resulting object needs to be initialized and the data returned by the database connector mapped into it. The mapping is done using XPath¹ function provided by server-side JavaScript API.

```
tw.local.person.firstName = tw.local.xmlNodeIterator.xpath('record/column[@name="
FIRSTNAME"]')[0].getText();
```

Apparently, this implementation approach has several drawbacks:

- It maps only the highest level of the Business Object attributes, not the nested complex ones.

¹XML Path Language

4. CASE STUDY

- It requires hard-coding of the Business Object attribute names, and also implementation details of the database schema (table structure, column types and names). That decreases the level of the flexibility and increases the effort needed to build and maintain such service.
- It provides only a single query to the table while having all the structure inscribed in the implementation. Implementing another SQL query (e.g. different set of columns, **WHERE** clause, ordering, etc.) to the same table would require to duplicate the code and possibility of inconsistency.
- The SQL statement is dependent on the implementation of the underlying database technology. Porting to another DBMS might introduce a substantial number of SQL statement edits required because of the differences between DBMS products [6].

4.3.1.2 Approach II - Reusable Asset

To deal with the observed problems, another solution was designed and implemented. The architectural model of the solution is shown in Figure 4.7. It consists of two parts:

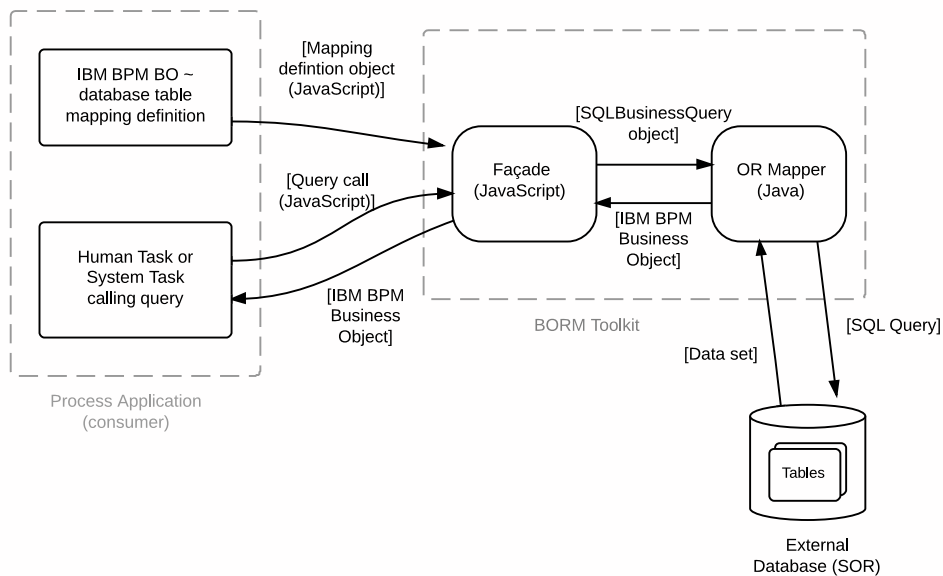


Figure 4.7: ORM Toolkit architecture

1. Object-relational mapper written in Java

This part performs the actual OR mapping. As an input, it takes `SQLBusinessQuery` object which contains all the information needed to execute the SQL query against the given database: database table name and corresponding Business Object name that should be returned, database column names and corresponding Business Object attributes, query parameters like `WHERE` clause, ordering, and related objects (in the given example, `Department` and `Address`). The structure of the object is shown in Figure 4.8. The implementation utilizes IBM BPM Java API to construct the output Business Object. The Java package is included in the Toolkit and the methods are called via standard Java Integration component.

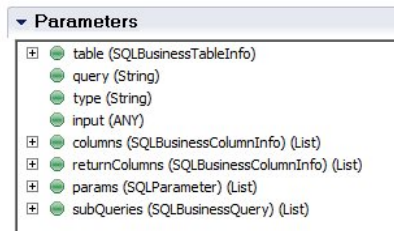


Figure 4.8: `SQLBusinessQuery` object structure

2. Façade written in server-side JavaScript

The JavaScript façade facilitates using of the Java ORM to the consumer of the Toolkit. It provides a way to define a mapping between the Business Objects and the database tables only once and then it is possible to query such structure dynamically. In our case, the definition of the mapping may look like Listing 4.1 and the query may be would be Listing 4.2. This kind of query can be called from any part of the process (any place where it is possible to use server-side JavaScript e.g. Human Task, or System Task) without worrying about the structure of the underlying database.

```

// create object models
var Person = new Borm();
var Dep = new Borm();

// map Person business object to PERSON table in database
Model.mapObject("Person")
.select("Person", "PERSON")
  // map Business Object parameters to DB table columns
  .param("personId", "PERSON_ID")
  .param("firstName", "FIRSTNAME")
  .param("lastName", "LASTNAME")
  // create relation - map column to subquery Department

```

4. CASE STUDY

```
.param("departmentId", "DEPARTMENT_ID").setRelation("department",
// new subquery with join column DEPARTMENT_ID
Dep.select("Department", "DEPARTMENT").where("DEPARTMENT_ID = ?")
.param("name", "NAME")
.param("headId", "HEAD_PERSON_ID").setRelation("head",
Head.select("Person", "PERSON").where("PERSON_ID = ?")
.param("personId", "PERSON_ID")
.param("firstName", "FIRSTNAME")
.param("lastName", "LASTNAME")
)
)
;
```

Listing 4.1: Mapping definition

```
tw.local.person = query.load("Person").where("lastName = 'Doe'").fetchSingle();
```

Listing 4.2: Query to the database

4.3.1.3 Comparison

The differences between the two solutions are depicted in Table 4.3.1.3. In terms of reusability, the most important characteristics that make the Approach II significantly more convenient is that the mapping is defined only once for each pair Business Object-database table. On top of that, it provides flexible API for querying the data source and it does not involve any SQL query composing. All the properties mentioned in the table significantly reduce amount and complexity of testing the implemented database data access. This testing may be very cumbersome and costly because it often involves many manual steps to be performed.

The important property of the reusable solution is that it facilitates changes in the data model. Since one of the key aspects of business process management is to provide the ability to change the business logic quickly, the data model may be a potential source of changes throughout the process life cycle. If the object-relational mapping is implemented in such rigid and code-duplicating style as in Approach I, the changes in the data model would introduce more development effort and thus prevent the desired level of agility.

4.3.2 Person Coach View

IBM Business Process Manager offers rich options for creating user interfaces through which the process participants may operate the tasks. The basic description how it works was stated in subsection 3.2.2.

One of the constraints given by IBM Process Designer is that it does not support inheritance and polymorphism of the Business Objects. This

Table 4.3: Comparison of ORM approaches

Characteristic	Simple	Custom ORM	Impact on reuse	Impact on
Mapping objects to tables	each query/service	once for a use case	high	creation, change
Data querying style	manual SQL query creation (based on database table knowledge)	custom built easy to use Javascript queries (based on Business Object knowledge)	medium	creation, change
Related tables linking	applicable, not very convenient	may be handled automatically	low	creation, change
Variability points	Data Source identifier, input variable	Data Source identifier, API for queries	high	creation, change, configuration
Usage style within IBM BPM	Service call	JavaScript method call	medium	creation, change
IBM BPM artifacts needed to be defined	1 Service per 1 database query	just 1 mapping service	medium	change, maintenance
DB platform dependency	high	low	medium	change
Error proneness	high	low	high	creation, change
Performance	lower	higher	medium	execution
DB connector type	built-in	custom written	none	configuration

4. CASE STUDY

implies that in some cases these principles need to be substituted by other techniques, like object composition. The main goal here is to have a number of Business Objects of different types that:

- share some common attributes
- are not created by duplicating the shared part for each type
- are presented on the Coach differently (accordingly to its type)
- still be substitutable for each other

The actual group of Business Objects that was dealt with was **Student**, **Employee**, and **ExternalPerson**. The problem is that these Business Objects need to be sometimes treated transparently as if it was a single **Person**. Thus, if there was inheritance available, it would have looked like in Figure 4.9. Instead, a composition hierarchy needed to be created in a following way (see Figure 4.10):

- **Student**, **Employee**, and **ExternalPerson** became children objects of common parent called **Person**
- **Person** was added a new attribute - **PersonType** which distinguished the type of parent
- Each subtype kept only attribute specific for it
- **Person** had always one and only one of subtype objects assigned, others were empty

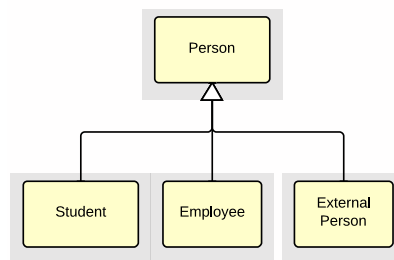


Figure 4.9: Person Object - Inheritance

A **Person** Business Object created this way was then used to create a **Person** Coach View which changes its appearance according to the subtype specified as an attribute of a **Person** variable bound to it. Therefore, the

4.3. Implementation of the Selected Reusable Assets



Figure 4.10: Person Business Obejct structure

interface of the Person Coach View asset remains the same for each the subtypes (it accepts a variable of type Person), and the appearance changes. Figure 4.11 shows the three different Person types bound to the same Coach View.

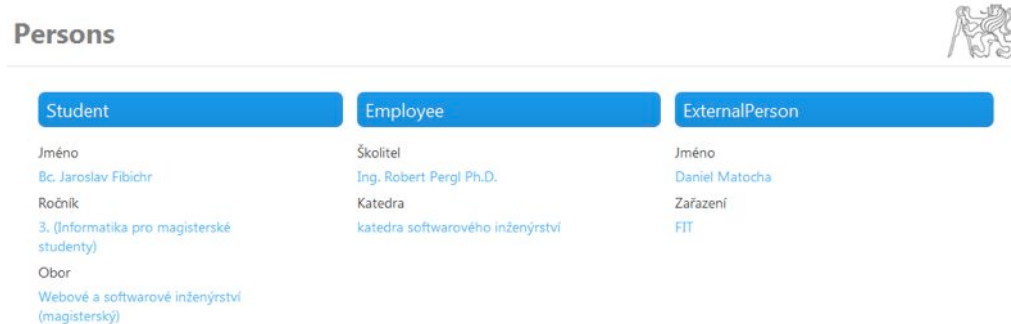


Figure 4.11:

Although the implementation of such Coach View is not very difficult, it provides great flexibility in use. In addition to the described functionality, the Person Coach view includes several options to configure the behaviour of the Coach View:

- **Person details (Dialog Box).** After clicking on the Persons name, a Dialog box with additional information like login or branch is shown.

- **Person view style.** Enables to choose between full view (with selected information according to the type) or name only.
- **Name label.** This option enables to change the label of the name (the default is "Name" but in different contexts it might be needed to change to e.g. "Examiner", or "Lecturer").
- **Name format.** Allows to show the name of the Person in different styles (with or without titles, leading by surname or first name), that might be used for example in lists when alphabetical order is convenient for sorting.
- **Empty person text.** Allows to define the text shown if no Person is bound to the Coach View (e.g. in a case when the Examiner is not determined yet).
- **Content Box.** The Coach View includes a Context Box to provide possibility to place additional Coach Views in it. This greatly increases the Coach View flexibility.



Figure 4.12: Person Coach View Configuration Options

The configuration section of the Coach View is shown in Figure 4.12. Only these values and the binding **Person** variable need to be set in order to use this Coach View. That makes this Coach View very easy to apply while being flexible enough for various use cases. Evaluation of impacts for the effort spent on creating such asset against building the functionality from scratch is presented in the section 5.1.

The **Person** Coach View is an example of domain-specific component. Its applicability is limited to the realm, that share the same requirements for how the Person should be represented. Because this Coach View is used by several Process Applications in our domain, it was moved to the separate Toolkit along with some other components providing such universal functionality. Since the Toolkit aggregates the artifacts that are specific to the CTU domain, it was called **CTU Toolkit**.

When extracting the artifacts to the Toolkit, there is one major challenge to face. The presented Coach View is tightly connected to the Business Object **Person**. This Business Object cannot be simply copied to the Toolkit. Instead, the underlying data model in the form of Business Objects should be extracted along with the Coach View to the Toolkit. A dependency on that Toolkit is then created in the Process Application, and the Business Objects from the Toolkit can be used just like they were in the Process Application itself. However, the developer needs to be very cautious with changes made on these externalized Business Objects because there might exist several such dependencies.

4.3.3 Change Log

The requirement for implementation of Change Log asset was based on practices performed at FIT as a part of AS-IS process activities. The procedure was found inconvenient for the participants due to excessive amount of paperwork.

The procedure of approving the diploma thesis topic consists of several steps. At the beginning, the thesis topic is discussed by a student and his/her supervisor. When they come up with a acceptable version, the topic is submitted to the approval process. For each topic, there are several individuals that need to review the thesis topic formulation. After reading the thesis topic proposal, he has two options - either approve the current version and send it to the next approval step, or write down the modification proposals and return it back to the previous approver. This procedure is repeated until the topic is approved by all the approvers on the path, or cancelled by the submitting supervisor. The thesis topic approval process is model is captured in Figure 4.13.

The important part is the exchange of the proposed version of the diploma thesis topic. In the AS-IS process, all the data was distributed on paper. Every time a modification of the topic was proposed by any of the approvers, a copy was made, and the changes were inscribed in the text. This way the last approver (the faculty dean) often ended up with a pile of different versions of the same diploma thesis topic. The reason for this technique was to enable tracking of the modifications made throughout the approval process and reduction of unnecessary interaction between the approvers.

Apparently, the way of exchanging the information was not very convenient for the participants. Therefore, for the TO-BE state of the process, the goal was to implement the tracking and viewing of modifications in the thesis topic more conveniently. And because this pattern was recognized in

4. CASE STUDY

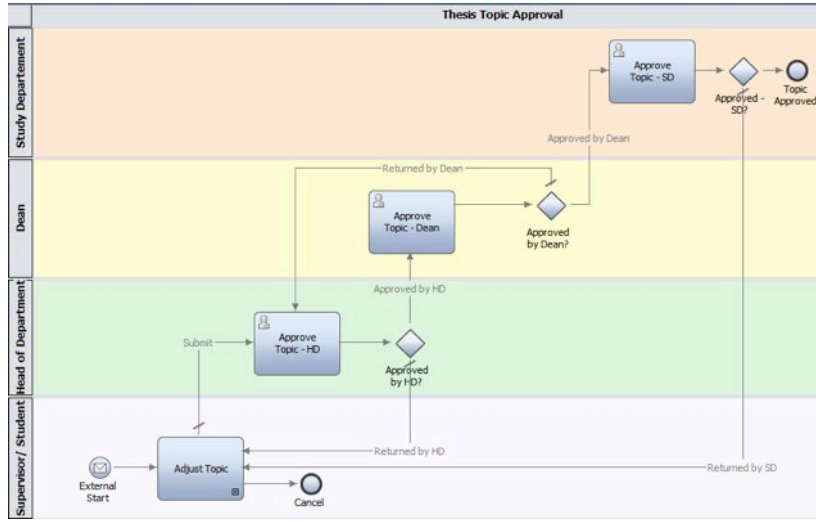


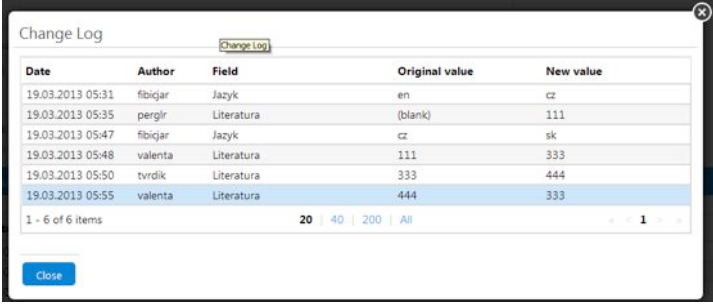
Figure 4.13: Thesis Topic Approval process - each of the human tasks require tracking of changes

some other processes that are intended for automation, this feature needed to be created in a reusable way.

The core of the implementation consisted of an algorithm, that enabled comparison of two instances of the same type of complex Business Object. The variable containing the information intended for tracking was then copied before the human task, and after its end were the two versions compared and changes stored in a specially defined structure. In our case, it was the **ThesisTopic** BusinessObject. It is important to take into account, that not all of the attributes of the Business Object needed to be tracked. Instead, the solution is required to enable to specify a subset of particular attributes, that should be tracked.

In the implemented solution, every time an approver wants to view the history of topic changes, he can access it through the Dialog box accessible from the Coach. An example of such history log is shown in Figure 4.14.

This functionality was decided to be implemented in a reusable manner. Although the main component, the comparer, was successfully developed to support any type of object (and therefore to facilitate reuse), the solution has one not negligible drawback, and that is the White Box style of reuse. That, in short, means that the number of steps to apply the asset in the target process is considerably higher than in the case of Glass Box or Black Box reuse. This results in lower reuse efficiency, as defined in subsection 2.3.1.



The screenshot shows a 'Change Log' dialog box with a table of changes. The table has five columns: Date, Author, Field, Original value, and New value. There are six rows of data. The last row is highlighted in blue. Below the table, there is a pagination bar showing '1 - 6 of 6 items' and a 'Close' button.

Date	Author	Field	Original value	New value
19.03.2013 05:31	fibiqjar	Jazyk	en	cz
19.03.2013 05:35	pergir	Literatura	(blank)	111
19.03.2013 05:47	fibiqjar	Jazyk	cz	sk
19.03.2013 05:48	valenta	Literatura	111	333
19.03.2013 05:50	tvrdik	Literatura	333	444
19.03.2013 05:55	valenta	Literatura	444	333

1 - 6 of 6 items 20 | 40 | 200 | All 1

Close

Figure 4.14: Changelog - Dialog Box for viewing the tracked changes

4.4 Experienced Issues

Throughout the BPM implementation projects, the overall development experience with the product was satisfactory. Although the graphical IDE remains simple to use, its capabilities cover most of the requirements for business process implementation tasks.

However, several problems were encountered; some of them complicated the development in general, others were closely related to practising reuse.

- **No JavaScript code search or refactoring.** The most significant shortcoming of the Process Designer is that although it provides a great server-side JavaScript API for handling various aspects of process execution within the executable process model, it does not provide any possibility to search through the Process Application for usages of Business Object or service references. That makes all the JavaScript code to be hard-coded within the artifacts and refactoring of complex Process Applications almost impossible to do.
- **Toolkit dependency removal issues.** Including a Toolkit in the Process Application needs to be a well-considered action. Once the Toolkit dependency is created and its artifacts are started to be widely used, it might be a cumbersome procedure to get rid of it. The product does not provide any way to list all the artifacts within the Process Application that use items from a particular Toolkit. On the other hand, if we forcefully remove the Toolkit dependency, the references to its parts become immediately invalid. And thus, if the entire Process Application is not checked for missing references, it might result in unexpected errors during the runtime. Additionally, the IDE slows down considerably when working with artifacts with missing references.

- **Reverting entire application.** Although the product provides a way to jump back in time to show the previous states of the Process Application, revert a single artifact, and even run new instances from the older versions, there is no convenient way to revert the entire Process Application. It must be done by complicated branching or exporting and importing the application.
- **Artifact dependencies.** During the reusable asset creation, the artifacts are usually moved from the Process Application to the external Toolkit to be reused. However, moving a larger set of artifacts between projects may cause lost references among the artifacts. Moreover, it is always necessary to consider dependencies that exist between domain-specific and general-purpose artifacts.
- **IDE performance and stability.** Although Process Designer is based on a powerful and stable Eclipse¹ IDE, its stability was occasionally inconsistent. Its concurrent development functionality requires constant, decent quality network connection to provide sufficient response times. In spite of its undisputed advantages, it prevents working on the project while staying offline.
- **Insufficient documentation.** Some parts of the product lacks an appropriate level of documentation provided by the vendor. This is the case mainly for JavaScript APIs and the most recently introduced features like the Coach Views and its interoperability with the rest of the system.

4.5 Summary

One of the of the main areas of the managerial development of the Faculty of Information Technology (FIT) is to introduce business process management practice into the organization. After the initial discovery and modelling phases, the first BPM projects entered the implementation phase in 2012.

Reuse played an important part in the BPM application development. In this chapter, I summarized the approach that was adopted throughout the implementation. An overview of Toolkits created or used within the projects was given, the reusable assets were categorized and the specific reuse styles discussed.

To depict the typical challenges of creating reusable components in the BPMS, three particular assets were selected and examined in detail. These

¹see <http://www.eclipse.org/>

assets are evaluated for their reuse qualities in the next chapter. The most significant issues encountered during the implementation activities were described.

Evaluation and Recommendations

In the previous chapter, several cases of using the reuse capabilities of IBM Business Process Manager were described. These cases were selected in such a way that it is possible to estimate the effort needed to develop these features. These estimations were used to achieve the main goal of this chapter which is to evaluate how efficient it is to implement a reusable asset in comparison to creating it from scratch every time when it is needed.

5.1 Measuring Procedure

5.1.1 Procedure and Conditions

The method to perform the estimations was discussed and described in section 5.1. In short, it consists of decomposing the implementation process into steps small enough to be able to estimate the time to finish them. Estimations for all the steps are then summed up and used as the values of variables in Equation 2.2. The formula provides a metric for determining the final results, and thus represents the achieved reusability rate. The reuse efficiency value directly depends on usage count of particular asset. Figure 5.1 clarifies the evaluation process represented in BPMN.

For each asset, to obtain the resulting reuse efficiency, I needed to analyse three implementation scenarios to acquire their total costs. Firstly, it was two ways to implement the given functionality:

1. **Reusable asset.** The feature is developed in a way, so that it can be easily reused on other place of the same (or another) Process Applica-

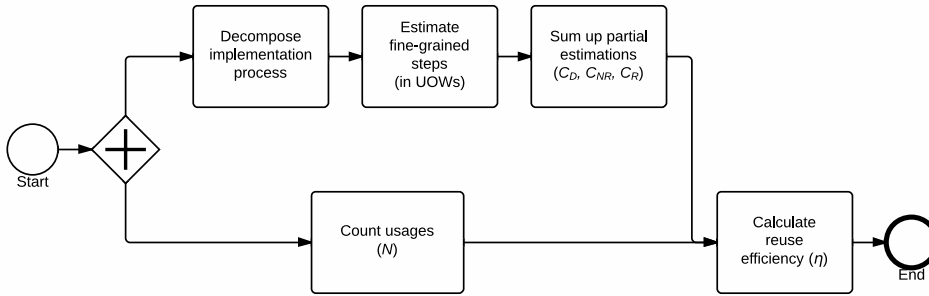


Figure 5.1: Reuse efficiency evaluation process

ation. Using of the reusable asset repeatedly should bring the desired effect of reduced resource requirements. However, the price for this is paid in increased upfront costs for more complicated implementation, more complex testing, packaging (e.g. extracting the asset into separate Toolkit), and documentation.

2. **Non-reusable development - always from scratch.** The feature is developed with no intention for reuse by design and it needs to be created from scratch every time when the particular functionality is required.

In addition to that, the process of **applying the reusable asset** needed to be estimated as well. These three estimations form in Figure 5.1 the upper part of the flow. To keep the estimation process model simple, it was modelled as a single path with no loops (that would consist of two iterations of development and one for application of reusable asset). To perform the estimations, several assumptions need to be made:

1. **Developer skills.** As the developers may vary in terms of their skills in particular area (business process implementation in this case), a reference model of developer needs to be specified. In our setting, the developers are software engineering students with basic knowledge of object-oriented principles, web technologies and related programming languages. The experience gained with the technology is on average 1 year.
2. **Translating units of work into person-hours.** The concept of unit of work (UOW) defined in subsection 2.3.2 was applied to the evaluation process, so that each implementation step is expressed

as a number of UOW. Given the reference model of developer described in previous paragraph, a correlation between UOW and the amount of actual work time may be done. Therefore, for this project, I defined the relationship between UOW and time simply as $1 \text{ UOW} \simeq 1 \text{ minute}$. This ratio appeared to be adequate because when total costs for the larger implementation parts were calculated, they equalled the number of minutes actually needed for the implementation. In this way, I was able to express the costs of implementation that was done using different technique like traditional coding in Java or writing documentation and include it into the evaluations.

3. **Averaging the cases.** Some assumptions needed to be made when estimating non-reusable development. Because the reusable asset usually forms a generalized solution for a feature used in different contexts, the costs for developing it in the particular case would be much lower. I used my experience in these cases to estimate the average case.

The description of functionality and approaches to the implementation of the selected assets was given in section 4.3.

5.1.2 Usage Counting

To calculate the resulting reuse efficiency, I also needed to count the *points of use* of the assets. These points are the places within the already implemented Process Applications where the particular asset was applied. In addition to that, I decided to include those places where the asset was not used but it could have been used and the only reason was that the reusable asset implementing the particular functionality was developed after the target Process Application (or its part that would use the feature). In addition, for the Doctoral Study Programme project, the usage counts were estimated on the basis of design (UML Class Diagrams, Coach mockups) because the implementation has not been done yet. However, this fact does not lower the reusability potential of the asset. The usage counts for the evaluated assets are shown in Table 5.1.

5.2 Results and Discussion

The estimations were acquired by following the steps of the three described scenarios (thus there are 9 tables in total). The performed steps were properly recorded in tables which are for the reader's convenience moved to

5. EVALUATION AND RECOMMENDATIONS

Asset name (Toolkit name)	ORM (ORM)	Person CV (CTU Toolkit)	Change Log (Change Log)
Final State Examination	34	14	0
Diploma Thesis	24	48	1
Doctoral Study Programme	20	40	2
TOTAL	102	78	3

Table 5.1: Usage count of reusable assets

the Table B. The tables contain implementation step descriptions, number of step counts, and the total cost estimated for the step. For each asset, I evaluated the acquired estimations and discussed their relevance to implementation procedure. Eventually, the reuse efficiency of the assets is calculated using the obtained estimation sum total.

5.2.1 Object-Relational Mapping

The object-relational mapping asset has some characteristics that needed to be considered during the evaluation. Since the Approach II (reusable) has been designed to separate the object-database mapping from the query calls, I had to estimate the asset usage part separately as well. The work decomposition for both approaches is shown in Table B.1 (Approach I - non-reusable development), Table B.2 (Approach II - reusable development), and Table B.3 (Approach II - application of reusable asset).

The asset usage evaluation (part IIb in Table B.2) introduced parameter k which denotes a number of different queries per one mapping. Although the value of this parameter could have been estimated based on intuition and experience with the usage, I preferred to examine the Process Applications to derive the value from the existing implementations. Table 5.2 shows the number of mappings, queries and the calculated ratio. The resulting average value was 3.09 different queries per 1 mapping, thus for the final reuse efficiency calculation, the parameter k was set to 3.

The estimation showed that the separation of mapping definitions and query calls might bring considerable reduction of work needed to access the data in the external database. While the straightforward, non-reusable solution needed 99 UOW for 3 different database queries ($k = 3$), the reusable asset provided the same functionality in 33 UOW.

Although the implementation of the reusable solution took considerable amount of time (more than 7000 UOW), the frequency of usage is so high

	Mappings count	Queries count	Ratio (k)
Final State Examination	14	7	2
Diploma Thesis	48	14	3.43
Doctoral Study Programme	40	12	3.33
TOTAL	102	33	3.09

Table 5.2: Derivation of mappings/queries ratio (k)

that it should be easily paid off. This assumption is supported by the fact that the asset's *scope* is general-purpose, independent to the domain.

Besides the development and usage of the asset, the completely changed approach to "define once, use endlessly" facilitates the ease of changes in the data model. That definitely further reduce the amount of development effort related to the ORM issue (although the estimation was not done for this case). For example, one changed Business Object attribute would lead to a single adjustment in the mapping definition when a reusable asset is used, instead of multiple changes in the SQL queries and explicit assignments in the non-reusable solution.

5.2.2 Person Coach View

Estimations of the Person CV asset are shown in Table B.4 (Approach I - non-reusable development), Table B.6 (reusable development), and Table B.5 (application of reusable asset).

Person Coach View is a representative of user interface artifact, that provide certain amount of predefined parameters to use. Although it is domain-specific, it is one of the most frequently used component (used 78 times so far) for viewing the data that was created (almost every Coach contained a person). This was augmented especially by making it "polymorphic".

The most important fact for asset that is used so frequently is the ease of use. That was successfully achieved, as the cost for using the component was estimated to only 3 UOW. By contrast, the estimated usage cost for building the average component for showing information about a person from scratch was 10 UOW. Again, as for the ORM, the reusable asset creation costs (342 UOW) should be paid off because of the usage count.

5.2.3 Change Log

Estimations of the Change Log asset are shown in Table B.7 (Approach I - non-reusable development), Table B.9 (reusable development), and Table B.8 (application of reusable asset).

Within the estimation, one parameter (denoted c in the tables) that emerged, could significantly influence the results. It was a number of places in the process where the changes of the tracked object could be made. In this case, I assumed intuitively that $c = 3$, because in average case, the approval process has three participants that may make the changes.

Although this asset provides domain independent functionality, some of its characteristics were negatively reflected in the estimations. The most important aspect was the reuse style of the asset, i.e. white box. The steps to apply the asset in the Process Application were estimated to 62 UOW, which was about half of the cost for developing the asset from scratch (125 UOW). That alone would not be a problem, if applicability of the asset was as high as it was for the previous cases. However, the core of the asset, the object comparer, might become a source for other functionalities created for Process Applications.

5.2.4 Reuse Efficiency

The final calculation of the reuse efficiency is shown in subsection 5.2.4. For each of the evaluated assets, three implementation processes costs were estimated using the proposed procedure. The only exception was made for Object-relational mapper which was written in Java and JavaScript. This implementation costs was derived from time reported by the developer – (120 hours = 7200 minutes). The usage counts were derived from implemented Process Applications (see Table 5.1). It showed that while the ORM and Person CV assets were very frequently applied functionality, the Change Log was too specific to be used so widely. This fact had direct impact on resulted reuse efficiency. To clarify the result a little more, I added the *Pay-off threshold* (N_1) value to the table.

Resulting values of reuse efficiency showed us, that in the first two cases (ORM and Person CV), the implementation of the reusable asset was worth the effort made. This applies even for the ORM asset even though the value did not reach its pay-off threshold. The reasons are described in the next paragraph. The only asset that was not worthwhile the effort according to the metric, was the Change Log asset. The main reason was probably given by the overall design immaturity which consisted of white box reuse style that require too many manual steps to apply the asset. Therefore, to

improve the reuse efficiency, more focus should have been given to better design, or the implementation simply should not have been done whatsoever. However, if some new usages for the asset emerge during the current or subsequent BPM projects, the Change Log asset implementation may eventually become worthwhile.

		ORM	Person CV	Change Log
Non-Reusable costs	C_{NR}	99	10	125
Reusable asset cost	C_R	33	3	62
Asset development costs	C_D	7297	342	582
Usage count	N	102	78	3
Pay-off threshold	N_1	111	49	10
TOTAL Non-Reusable	NC_{NR}	10098	780	375
TOTAL Reusable	$NC_R + C_D$	10663	576	768
Reuse efficiency	η	0.95	1.35	0.49

Table 5.3: Reuse efficiency calculation for evaluated assets

Besides the results that were quantified, I conclude several other findings that I observed and that affects the overall reuse efficiency of the assets:

1. **Avoiding duplications.** One of the merits of software reuse is reducing duplication of work. That was also proven during our projects. Instead of cumbersome searching and refactoring of the code and artifacts, the change is done only once. That indirectly improves the achieved reuse efficiency.
2. **Assets are evolving.** Over the time, not many of the implemented assets remained the same forever. As the emerging potential consumers may have varying requirements, the asset sometimes needs to be modified in order to meet them. Development of assets should be done in accordance to the open/closed principle¹, therefore the asset grows and the consumers use only a subset of its functionality. On the one hand, this may provide the consumer some flexibility in use, on the other hand, the component may become too complex.

¹”software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification” [51]

5. EVALUATION AND RECOMMENDATIONS

3. **Evolution costs.** The changing aspect described in previous paragraph has considerable impact on the reuse efficiency metric that I used for the evaluation. From the experience gained during the projects I have assumed that, each modification made to the asset causes additional costs that need to be taken into account in the efficiency evaluation model. Figure 5.2 (derived from Figure 2.7) schematically illustrates how the pay-off threshold might be moved if we count with the costs of changes made to the assets. For some cases where they are too high, the reuse might never pay off (the lines in the graph never cross). I find this factor as a considerable risk to the reuse.

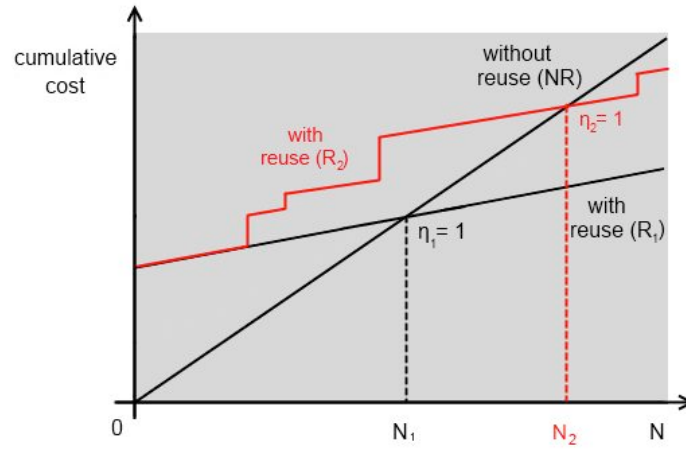


Figure 5.2: Impact of asset changes on reuse efficiency. R_1 denotes reusable asset without changes; R_2 denotes reusable asset with changes; NR denotes non-reusable implementation; N denotes number of reuses; η_1, η_2 are the points where the reuse becomes to pay off

4. **Relevancy of the results.** The selection of the evaluation cases were done in order to demonstrate assorted approaches to implementation of reuse in IBM BPM. However, the results and conclusions may have been different if the asset selection was chosen differently, or the evaluation done on larger set of assets. That might be achieved by some sort of automatic estimation. However, the only viable possibility to do that in IBM BPM would be to employ a data mining methods on process metadata stored in Process Center database.

5.2.5 Comparison to Empirical Data

In the previous section, measuring results gave us certain notion about efficiency of the reuse that was achieved in the evaluated cases. However, it would be interesting to compare the measured costs with some empirically acquired data. Jacobson et al. [36] presented several guidelines (described in subsection 2.3.3), that apply generally to software reuse and that I used as referential data to compare my results to, see Table 5.4.

		ORM	Person CV	Change Log	Jacobson et al.
Non-reusable costs	C_{NR}	99	10	125	
Reusable asset cost	C_R	33	3	62	
Asset development costs	C_D	7297	342	582	
Pay-off threshold	N_1	111	49	10	$\langle 3, 5 \rangle$
Usage costs ratio	C_R/C_{NR}	0.33	0.30	0.50	0.25
Creation costs ratio	C_D/C_{NR}	73.71	34.20	4.66	$\langle 1.5, 3 \rangle$

Table 5.4: Comparison of results with findings of Jacobson et al. [36]

We can see that only *Usage cost ratios* of our reusable solutions are somewhat close to the value proposed by Jacobson. In those other metrics, the values differed in order.

Reasons for these findings are various. Firstly, the IBM BPM platform has its specifics that resulted in much faster creation of non-reusable feature then a reusable one. Secondly, the general programming skills of developers creating the assets were not on the level of an average professional (as it was in Jacobson's case), and also sometimes an insufficient documentation of the product lead to a longer time of utilization of the more advanced techniques needed to create the reusable assets. Finally, the granularity of the assets may have been much higher in our case. However, even though the pay-off threshold was set quite high, the very high applicability of the assets we produced has paid off (as shown in the previous section).

5.3 Recommendations

Based on the experience with implementations of Process Applications gained during the BPM projects, I have put together a list of best practices and recommendations that could be used as a reference for subsequent work. This list involves primarily practices that might increase the degree

5. EVALUATION AND RECOMMENDATIONS

of reuse and overall efficiency of BPM applications development. It can ultimately provide the process owners, participants and BPM project sponsors the added value that forms the merit of all BPM initiatives.

1. **Do not repeat yourself (or anybody else).** In many occasions, when a feature that is considered to be created, is already existing in some form. Prior to the actual start of designing, modelling or coding, search the available resources that might contain if not the same than at least similar functionality. This way you can deliver the desired feature in a fraction of the time possible spent on the developing it from scratch.
2. **Keep it simple.** The best type of asset a developer can provide is a one that can be very easily applied. Adding too many extra features might unnecessarily bloat the reusable asset and make it harder to use and sometimes even discourage the potential consumers from applying it at all.
3. **Test thoroughly.** Before an asset is claimed as ready, make sure that you thought of every single use case that might be considered by the potential asset consumer. Create services or scripts that can be used for testing the asset repeatedly to save time after making modifications.
4. **Show off with success.** Whenever you manage to achieve anything of worth, do not hesitate to expose it to the team. It has been proven to be useful to organize regular technical meetings on which each team member that had succeed to develop anything potentially beneficial for the others briefly refers about it or demonstrates it to make the others aware.
5. **Share.** When you think the achievement might be beneficial even outside the team, publish it in the community. That may be of great use not only to its members but also to the asset producer, as he gains the feedback on what might be done better or how can be the asset extended.
6. **Leave trails.** When you make change to any existing asset, make sure that the modification does not limit the original functionality. By all means, it is always a good practice to fully describe what was changed and how it influences the behaviour or usage of the asset.
7. **Tagging is good.** IBM Process Designer has several capabilities to make the development with usage of Toolkits much more efficient. For

example, define a set of Tags that would help to search the assets for others. Over some time, even a small team might produce hundreds of artifacts that would eventually turn into confusing stack of assets failed to be found.

8. **Teamwork is everything.** Leverage the potential of the skills mixture within the team. Many times when you struggle with a task, there might be someone else who would handle it in no time and vice versa.
9. **Think twice.** Examine outcomes of the process modelling and analysis stages to decide whether the particular function is a good candidate for producing a reusable asset. Even if it is more than obvious that it is a good idea, sometimes it is just a waste of time that might be spent on other, more pressing tasks.
10. **Communicate.** Sharing ideas and experienced problems is just one side of the deal. The power of mutual motivation and maintaining a pleasant working atmosphere may increase the efficiency of the team just as well.

5.4 Summary

The specific approach to the process development in IBM BPM provides decent options for reusing the features that were created previously. These reusable assets, packaged in Toolkits, can be reused in subsequent projects to facilitate the development process. The goal of this chapter was to apply the metric proposed in chapter 2 to evaluate the efficiency of the asset reuse.

The estimation of development costs was performed by a decomposition of the work process into measurable steps. Summing them up gave us quantifications that could be used for calculation of the results.

The results has shown that even if the reusable asset development is very time-demanding, it may bring considerable savings in the future. In comparison with creating the particular functionality from scratch every time it is needed, the reusable asset needs to be very easy to apply (preferably as Black Box), otherwise it needs to be applied rather frequently to pay off. Typically, the general-purpose assets were assumed to be more applicable and thus more reuse efficient than domain-specific assets. However, due to the specific selection of the evaluated assets, did not show.

5. EVALUATION AND RECOMMENDATIONS

Reuse efficiency of an asset is decreasing if it requires any additional modifications in order to meet the emerging requirements of the new potential consumers.

Conclusion

The aim of this thesis was to analyse the options for leveraging reuse within the BPM application development. To achieve this objective, I defined three partial goals.

The first goal was accomplished by a literature search followed by a detailed description of the two major topics. In the first chapter, I focused on providing a detailed overview of BPM-related principles that would serve as a context for application of reuse in BPM application development. That was done by description of history, present, methodology, and the most important standard associated with BPM, as well as its relation to SOA. The second chapter elaborated on issues of software reuse that aimed on aspects applicable to BPM implementations. That included categorization of reusable assets, the their life cycle, form of their description, storing, searching and sharing. In another part of the second chapter, I described possible measurements of reuse, derived a metric for reuse efficiency and proposed a method for the asset development cost estimation.

The second goal defined for this thesis was to analyse the selected BPM suites with regards to the reuse capabilities. Although the three reviewed products (Bizagi, Activiti, IBM BPM) aim at similar target customers, their approaches to the process application development and reuse varies widely. Out of the three, IBM BPM was the only one solution that offered an easy to use process implementation without the need of extensive coding and features enabling cross-application reuse. The reuse potential is mainly in rich user interfaces, backend services and also shared asset repository. The reuse in IBM BPM is supported not only technically, but also by methodology associated with the product.

The third goal of the thesis was defined to verify the assertion that a wide utilization of reuse in IBM BPM is efficient on long-term basis. To

fulfil the goal I conducted a case study within the ongoing BPM projects at FIT. I identified three different features that were possible to develop in reusable and non-reusable way and analysed these approaches in detail. The evaluation consisted of the specially defined cost estimation method based on decomposition of the implementation steps.

The results showed, that reuse in IBM BPM can be leveraged very efficiently when certain conditions are met. First of all, evidently the most reuse-efficient are those assets that are very easy to apply and that are widely applicable. Secondly, even creation of the reusable asset that is very expensive (in terms of development costs) might pay off. Thirdly, the assets that are applied in a white-box style (therefore the consumer needs to do some modifications to use the asset) should be carefully considered to be implemented as reusable assets. However, to conclude the evaluations, I can say that creating reusable may be worthwhile despite its increased development costs.

This work has **several limitations**. Firstly, the simplified method for implementation costs estimation might have distorted the resulted values. Secondly, the number of the evaluated assets was too small to conclude with statistic certainty.

While composing of this thesis, **I have learned** a plenty of new information about BPM-related methodologies, implementation techniques and products. Also, elaborating on the software reuse, which was the other topic of my thesis, has greatly broadened my knowledge, and inspired me to apply some of the findings to practice. I sincerely believe that all this experience will help me to succeed in my subsequent career in the information technology field.

This **work may be extended** in several ways. First of all, there is the gathering of the data from the projects that, if evaluated, would result in much more precise results. Another option would be building an automatic data collector. That would be necessary for large-scale evaluations of reuse efficiency since the method used in this work is quite cumbersome. And thirdly, implementing a comprehensive collection of reusable assets to facilitate the fastest possible BPM application development, which was already started during the ongoing BPM projects.

This **work may be used** by all the IBM BPM application developer teams and individuals that are not sure if their reusable development efforts will eventually pay off. In addition to the goals defined at the beginning, I have put together a set of guidelines for successful reuse practice in a small BPM development team (see section 5.3) and a list of issues that I encountered during the implementations that hinder efficient BPM project developments leveraging reuse (see section 4.4).

Bibliography

- [1] Aalst, W. M.; van Dongen, B. F.; Herbst, J.; etc.: Workflow mining: a survey of issues and approaches. *Data & knowledge engineering*, volume 47, no. 2, 2003: pp. 237–267.
- [2] Abrams, C.; Schulte, R. W.: Service-Oriented Architecture Overview and Guide to SOA Research. <http://www.gartner.com/id=575608>, January 2008, [online].
- [3] Ackerman, L.; Gonzalez, C.: Patterns-Based Engineering. <http://patternsbasedengineering.net/>, November 2011, [online, accessed 01/04/2013].
- [4] Activiti.org: Activiti BPM Platform. <http://www.activiti.org/>, [online, accessed 02/04/2013].
- [5] Arsanjani, A.: Service-oriented modeling and architecture. <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/>, November 2004.
- [6] Arvin, T.: Comparison of different SQL implementations. <http://troels.arvin.dk/db/rdbms/#insert-multiple>, [online, accessed 20/04/2013].
- [7] Association of Business Process Management Professionals; Binner, H.; Bariff, M.; etc.: *Business Process Management Common Body of Knowledge*. Association of Business Process Management Professionals, 2009, ISBN 9781442105669.
- [8] Atego: Atego Asset Library. <http://www.atego.com/products/atego-asset-library/>, [online, accessed 17/04/2013].

BIBLIOGRAPHY

- [9] Behara, G. K.: BPM and SOA: a strategic alliance. *BP Trends*, , no. May, 2006.
- [10] Bizagi: Bizagi. <http://www.bizagi.com/>, [online, accessed 13/03/2013].
- [11] Brown Alan, W.; Wallnau Kurt, C.: A Framework for Systematic Evaluation of Software Technologies. *IEEE Software*, September, 1996.
- [12] Center for Systems and Software Engineering: COCOMO II. http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html, [online, accessed 13/04/2013].
- [13] Chinosi, M.; Trombetta, A.: BPMN: An introduction to the standard. *Computer Standards & Interfaces*, volume 34, no. 1, 2012: pp. 124 – 134, ISSN 0920-5489.
- [14] Clements, P.; Northrop, L.: *Software product lines: practices and patterns*. SEI Series in Software Engineering Series, Addison Wesley Professional, 2002, ISBN 9780201703320.
- [15] Consortium, S. P.: *Reuse Adoption Guidebook*. Software Productivity Consortium, 1992.
- [16] Das, M.; Deb, M.; Wilkins, M.: *Oracle Business Process Management Suite 11g Handbook*. McGraw-Hill Education, 2011, ISBN 9780071754507.
- [17] Davenport, T.: *Process innovation: reengineering work through information technology*. Harvard Business School Publishing India Pvt. Limited, 1993, ISBN 9780875843667.
- [18] DeCarlo, J. W.; Ackerman, L.; Elder, P.; etc.: *Strategic Reuse with Asset-Based Development*. IBM, International Technical Support Organization, 2008.
- [19] Dixon, J.: Hype Cycle for Business Process Management, 2012. <http://www.gartner.com/id=2096519>, July 2012, [online, accessed on 01/03/2013].
- [20] Durvasula, S.; Guttman, M.; Kumar, A.; etc.: SOA Practitioners' Guide Part 1: Why Services-Oriented Architecture? Sept. 2006: pp. 1–18.

- [21] Foundation, T. D.: Dojo Toolkit. <http://dojotoolkit.org/>, [online, accessed 04/04/2013].
- [22] Frakes, W. B.; Kang, K.: Software reuse research: Status and future. *Software Engineering, IEEE Transactions on*, volume 31, no. 7, 2005: pp. 529–536.
- [23] Gartner, Inc.: The Gartner Research Process And Methodologies. <http://imagesrv.gartner.com/research/methodologies/methodologies.pdf>, 2011, [online, accessed on 02/03/2013].
- [24] Giaglis, G.: A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques. *International Journal of Flexible Manufacturing Systems*, volume 13, no. 2, 2001: pp. 209–228, ISSN 0920-6299.
- [25] Gillot, J.-N.: *The Complete Guide to Business Process Management*. Lulu.com, 2008, ISBN 9782952826624.
- [26] Hammer, M.; Champy, J.: *Reengineering the Corporation: Manifesto for Business Revolution*, A. HarperCollins, 2009, ISBN 9780061808647.
- [27] IBM: IBM Blueworks Live. <https://www.blueworkslive.com/>, [online, accessed 02/02/2013].
- [28] IBM: IBM BPM Community Wiki. <http://bpmwiki.blueworkslive.com/display/Dashboard/HOME>, [online, accessed 24/04/2013].
- [29] IBM: IBM Business Process Manager V8.0.1 information center. <http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp>, [online, accessed 04/04/2013].
- [30] IBM: IBM developerWorks - IBM Business Process Manager. <https://www.ibm.com/developerworks/forums/forum.jspa?forumID=2382>, [online, accessed 24/04/2013].
- [31] IBM: IBM Rational Asset Manager. <http://www.ibm.com/developerworks/rational/products/ram/>, [online, accessed 17/04/2013].
- [32] IBM Redbooks: Creating a BPM Center of Excellence (CoE). <http://www.redbooks.ibm.com/abstracts/redp4898.html>, [online, accessed 15/03/2013].

- [33] IBM Redbooks: The Process Architect: The Smart Role in Business Process Management. <http://www.redbooks.ibm.com/abstracts/redp4567.html>, [online, accessed 15/03/2013].
- [34] IBM Redbooks: *Scaling BPM Adoption from Project to Program With IBM Business Process Manager*. IBM redbooks, Vervante, 2011, ISBN 9780738436081.
- [35] Institute, P. M.: *A guide to the project management body of knowledge (PMBOK guide)*. Guide to the Project Management Body of Knowledge, 2000, Project Management Institute, 2000, ISBN 9781880410257.
- [36] Jacobson, I.; Griss, M.; Jonsson, P.: *Software reuse: architecture process and organization for business success*. ACM Press books, ACM Press, 1997, ISBN 9780201924763.
- [37] Jensen, C. T.; Rob High, J.; Mills, S.: Achieving business agility with BPM and SOA together. <ftp://public.dhe.ibm.com/common/ssi/ecm/en/wsw14078usen/WSW14078USEN.PDF>, October 2009, [online, accessed 04/02/2013].
- [38] Jeston, J.; Nelis, J.: *Business Process Management: Practical Guidelines to Successful Implementations*. Butterworth-Heinemann, 2006, ISBN 9780750669214.
- [39] Jorgensen, M.; Shepperd, M.: A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions on*, volume 33, no. 1, 2007: pp. 33–53.
- [40] Kemsley, S.: A Short History of BPM. <http://www.column2.com/category/bpmhistory/>, May 2006, [online].
- [41] Kemsley, S.: Enterprise BPM Webinar Q & A Followup. <http://www.column2.com/2011/11/enterprise-bpm-webinar-qa-followup/>, November 2011, [online].
- [42] Ko, R. K.; Lee, S. S.; Lee, E. W.: Business process management (BPM) standards: A survey. *Business Process Management Journal*, volume 15, no. 5, 2009: pp. 744–791.
- [43] Kolban, N.: Kolban’s Book on IBM BPM. <http://www.neilkolban.com/IBM/>, [online, accessed 01/04/2013].

- [44] Koster, S. R.: *An evaluation method for Business Process Management products*. Master thesis, Business Information Technology, University of Twente, May 2009.
- [45] Krafzig, D.; Banke, K.; Slama, D.: *Enterprise SOA: Service-Oriented Architecture Best Practices*. The Coad Series, Prentice Hall Professional Technical Reference, 2005, ISBN 9780131465756.
- [46] Krueger, C. W.: Software reuse. *ACM Comput. Surv.*, volume 24, no. 2, June 1992: pp. 131–183, ISSN 0360-0300.
- [47] Larsen, G.: Asset Based Development. <http://xml.coverpages.org/Larsen-RAS200311.pdf>, November 2003, [online, accessed 31/03/2013].
- [48] Larsen, G.: Model-driven development: Assets and reuse. *IBM Systems Journal*, volume 45, no. 3, 2006: pp. 541–553.
- [49] Lindsay, A.; Downs, D.; Lunn, K.: Business processes-attempts to find a definition. *Information and Software Technology*, volume 45, no. 15, Dec. 2003: pp. 1015–1019, ISSN 09505849.
- [50] List, B.; Korherr, B.: An evaluation of conceptual business process modelling languages. In *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, 2006, pp. 1532–1539.
- [51] Meyer, B.: *Object-oriented software construction*. PRENTICE-HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE, Prentice Hall PTR, 1997, ISBN 9780136291558.
- [52] Object Management Group: Reusable Asset Specification. <http://www.omg.org/spec/RAS/>, November 2005, [online, accessed 04/04/2013].
- [53] Object Management Group: Business Process Model And Notation. <http://www.bpmn.org/>, January 2011, [online, accessed 04/02/2013].
- [54] Osmani, F.: System-of-Record Architecture for Process-Driven Solutions. <http://soapower.com/IBMBPM/Whitepapers/IBM-BPM-BP-BestPractices-SystemOfRecord-BPM-SOR-Architecture.pdf>, [online, accessed 22/04/2013].
- [55] Prieto-Díaz, R.: Status report: Software reusability. *Software, IEEE*, volume 10, no. 3, 1993: pp. 61–66.

- [56] Rademakers, T.; Baeyens, T.; Barrez, J.: *Activiti in Action: Executable Business Processes in BPMN 2.0*. Manning Pubs Co Series, Manning Publications Company, 2012, ISBN 9781617290121.
- [57] Richardson, C.; Miers, D.: The Forrester WaveTM: BPM Suites, Q1 2013. <http://www.forrester.com/The+Forrester+Wave+BPM+Suites+Q1+2013/fulltext/-/E-RES88581>, March 2013, [online, accessed on 15/03/2013].
- [58] Ruževičius, J.; Milinavičiūtė, I.; Klimas, D.: Peculiarities of the business process management lifecycle at different maturity levels: The banking sector's case. *ISSUES OF BUSINESS AND LAW*, volume 4, 2012: pp. 69–85, ISSN 2029-9214.
- [59] Sametinger, J.: *Software Engineering with Reusable Components*. Springer, 1997, ISBN 9783540626954.
- [60] Schulte, R. W.; Abrams, C.: The Business Impact of Service-Oriented Architecture. <http://www.gartner.com/id=570808>, December 2007, [online].
- [61] Silver, B.: *Business Process Model and Notation Method and Style*. Cody-Cassidy Press, 2011, ISBN 9780982368114.
- [62] Smith, H.; Fingar, P.: *Business Process Management: The Third Wave*. Advanced business-technology books for competitive advantage, Meghan-Kiffer Press, 2003, ISBN 9780929652337.
- [63] Snabe, J. H.; Rosenberg, A.; Møller, C.: *Business Process Management: The SAP Roadmap*. SAP PRESS, 2008, ISBN 978-1592292318.
- [64] Vaníček, J.; Česká zemědělská univerzita v Praze. Provozně ekonomická fakulta: *Měření a hodnocení jakosti informačních systémů*. Credit, 2000, ISBN 9788021306677.
- [65] Vejražková, Z.: *Business Process Modeling and Simulation: DEMO, BORM and BPMN*. Master thesis, Faculty of Information Technology, Czech Technical University in Prague, January 2013.
- [66] Voříšek, J.; Basl, J.; Vysoká škola ekonomická v Praze: *Principy a modely řízení podnikové informatiky*. Oeconomica, 2008, ISBN 9788024514406.

Acronyms

ABD	Asset-Based Development
API	Application Programming Interface
BAM	Business Activity Monitoring
BO	Business Object
BPD	Business Process Definition
BPM	Business Process Management
BPMN	Business Process Modeling Notation
BPMS	Business Process Management System/Suite
BPR	Business Process Reengineering
BRMS	Business Rules Management System
CTU	Czech Technical University
CV	Coach View
DB	database
DDSP	Doctoral Degree Study Programme
ERP	Enterprise Resource Planning
FIT	Faculty of Information Technology
IBM	International Business Machines

A. ACRONYMS

IBM BPM IBM Business Process Manager

IDE Integrated Development Environment

KPI Key Performance Indicator

LDAP Lightweight Directory Access Protocol

OMG Object Management Group

ORM Object-relational mapping (or mapper)

PA Process Application

PC IBM Process Center

PD IBM Process Designer

PS IBM Process Server

RAS Reusable Asset Management

REST Representational State Transfer

SLA Service Level Agreement

SOA Service-oriented architecture

SQL Structured Query Language

UI User Interface

UML Unified Modeling Language

UOW Unit(s) of Work

URI Uniform Resource Identifier

WBS Work Breakdown Structure

WSDL Web Services Description Language

XML eXtensible Markup Language

XSLT Extensible Stylesheet Language Transformations

APPENDIX **B**

Development Cost Estimations

Step description	item count	est. UOWs	hard -coded attributes	remarks
Create Service	1	1	-	
Place and link service components	4	2	-	
Define input, output, private variables	5	2	-	
Define validation condition for gateway	1	1	-	most commonly 1 but may be more
Compose SQL query	1	5	BO, DB	varies from 1 for simple SELECT, to several UOWs for complex INSERTs or UPDATEs
Attach and config DB Connector service	1	1	-	
Design exception handling	2	1	-	
Map returned data set to Business Object	1	10	DB, BO	varies depending on complexity of returned data set and resulting BO
Create test cases	2	4	DB, BO	varies depending on complexity of mapped BO, DB table, and query conditions
Run tests	2	6	DB, BO	
TOTAL (I) * k		33 * k		(k denotes number of different queries per mapping)
TOTAL (I), k=3		99		

Table B.1: Evaluation of ORM - Approach I (non-reusable development)

B. DEVELOPMENT COST ESTIMATIONS

Step description	item count	est. UOWs
Design, code (Java, JavaScript), testing - 100 man-hours		7200
Create Model Mapper template	1	20
Package	17	17
Documentation		60
TOTAL		7297

Table B.2: Evaluation of ORM - Approach II (development of reusable asset)

IIa / Step Description	item count	est. UOWs	hard -coded attributes	remarks
Define mapping	1	5	BO, DB	varies depending on complexity of mapped BO, DB table
Create test cases	2	4	BO, DB	
Run tests	2	6	BO, DB	
TOTAL (IIa)		15		
IIb / Step Description	item count	est. UOWs	hard -coded attributes	remark
Create query	1	$k * 2$	BO	varies on complexity of query conditions
Create test cases	2	$k * 2$	DB, BO	
Run tests	2	$k * 2$	DB, BO	
TOTAL (IIb)		$k * 6$		
TOTAL (IIa + $k * \text{IIb}$)		$15 + k * 6$		$(k$ denotes number of different queries per mapping)
TOTAL (II), $k=3$		33		

Table B.3: Evaluation of ORM - Approach II (application of reusable asset): (IIa) shows evaluation of mapping definition, (IIb) shows evaluation of query calls

Step description	item count	est. UOWs
Create section layout and place elements	3	2
Set labels and bind variables	3	1
Configure elements, set visibility	3	2
Code dynamic behaviour (mostly visibility)	1	2
Debug and test the Coach	1	3
TOTAL		10

Table B.4: Estimation of Person CV - non-reusable development

Step description	item count	est. UOWs
Place element	1	1
Bind variable	1	1
Configure element	1	1
TOTAL		3

Table B.5: Estimation of Person CV - application of asset

Step description	item count	est. UOWs
Design		30
Create CV	1	1
Create section layout and place elements	27	18
Set labels and bind variables	27	9
Configure elements, set visibility	27	18
Create configuration options and related types	4	8
Code dynamic behaviour (100 lines of JavaScript code)	100	200
Debug and test the Coach template	1	3
Test		30
Package	5	5
Documentation		20
TOTAL		342

Table B.6: Estimation of Person CV - development of reusable asset

Step description	item count	est. UOWs
Create Wrapper Service	3	$c * 15$
Code Manual comparisons in JavaScript	20	40
Create Change Log Coach	1	10
Debug and test solution		$c * 10$
TOTAL		$50 + c * 25$
TOTAL ($c = 3$)		125

Table B.7: Estimation of Change Log - non-reusable development; c denotes number of activities in BPD with tracked changes

B. DEVELOPMENT COST ESTIMATIONS

Step description	item count	est. UOWs
Place, bind, and configure Change Log CV	1	3
Copy wrapper template	1	1 * c
Adjust wrapper (variable creation and mapping)	1	15 * c
Create Changelog variable in BPD	1	1
Create Change Log configuration	1	10
TOTAL		14 + 16 * c
TOTAL (c = 3)		62

Table B.8: Estimation of Change Log - application of asset; c denotes number of activities in BPD with tracked changes

Step description	item count	est. UOWs
Design		100
Create Business Object	1	3
Create Services	3	3
Code, debug and test comparison algorithm (150 lines of JavaScript)	150	300
Create Human Service template wrapper (4 components, variable mapping, JavaScript coding)	4	18
Create Change Log Dialog Box CV	1	10
Create Coach Template	4	8
Create Change Log Configuration template	1	10
Test		60
Package	10	10
Documentation		60
TOTAL		582

Table B.9: Estimation of Change Log - development of reusable asset

Contents of enclosed CD

	readme.txt	the file with CD contents description
	src	the directory of source codes
	thesis	the directory of L ^A T _E X source codes of the thesis
	text	the thesis text directory
	thesis.pdf	the thesis text in PDF format
	thesis.ps	the thesis text in PS format
	twx	the directory with process application archives
	FSE.twx	the exported Final State Examination Process Application
	DTH.twx	the exported Diploma Thesis Process Application
	DSP.twx	the exported Doctoral Study Program Process Application