

# MASTERARBEIT

## Multiobjective Decision Support for Selecting Business Application Portfolios

Ausgeführt am **Institut für Softwaretechnik und Interaktive Systeme**  
der **Technischen Universität Wien**

unter Anleitung von **o.Univ.Prof. Dipl.-Ing. Dr. techn. A Min Tjoa**

durch **Jan Pichler,**  
**Tamussinostr. 16/2/1, A-2340 Mödling,**  
**Matrikel-Nr. 9800843**

Wien am 01.05.2007

.....  
Jan Pichler

.....  
Prof. DI Dr. A Min Tjoa

## **Acknowledgements**

I would like to thank many people for their unremitting support during this work and the rest of my studies. All you who have helped me, allow me to express my appreciation and my deepest thanks for your understanding and your care. In particular I want to thank my marvellous girlfriend, my wonderful family and my dear Markus for believing in me (and proof-reading this work).

I also want to thank Prof. DI Dr. A Min Tjoa for allowing me to write about this interesting and challenging piece of software management theory and DI Mag. Thomas Neubauer for leading my way with scientific expertise and professional advice.

## **Zusammenfassung**

Der geschäftliche Erfolg vieler Unternehmen wurde in den letzten Jahren immer mehr von der Funktionalität und Effizienz ihrer IT-Systeme abhängig. Dennoch erscheint vielen Unternehmen eine strukturierte Auswahl von Software zur optimalen Unterstützung aller Geschäftsprozesse nach wie vor zu kompliziert, da bis dato nur wenige geeignete Methoden zur strukturierten Bewertung und Auswahl von Unternehmenssoftware existieren. Hinzu kommt dass bei der Analyse von Software-Portfolios nicht nur die Charakteristika einzelner Applikationen sondern auch Abhängigkeiten zwischen den unterschiedlichen Applikationen berücksichtigt werden müssen, was nur wenige der momentan gebräuchlichen Methoden erlauben.

Das in dieser Arbeit entwickelte Modell basiert auf dem MODS (Multiobjective Decision Support) Ansatz und ermöglicht eine automatische Zusammenstellung Pareto-effizienter Software-Portfolios die in ihren Charakteristika zuvor definierten Gültigkeitsbedingungen genügen. Mit Hilfe eines interaktiven Analyse-Werkzeuges kann der so entstandene Lösungsraum schrittweise reduziert werden bis nur noch eine geringe Anzahl optimaler Lösungen verbleibt.

Durch die unkomplizierte und zeitoptimale Evaluierung beliebiger Software-Portfolios ermöglicht dieses Modell Unternehmen erstmals sowohl eine effiziente Auswahl von IT-Komponenten als auch eine regelmäßige Analyse und Optimierung bereits bestehender IT-Infrastrukturen.

# Contents

<b>1</b>	<b>Abstract</b>	<b>6</b>
<b>2</b>	<b>Introduction</b>	<b>8</b>
2.1	Motivation . . . . .	8
2.1.1	IT Management Approaches . . . . .	9
2.1.2	Valuation of IT Resources . . . . .	9
2.2	Research Goals . . . . .	10
<b>3</b>	<b>Related Work / Fundamentals</b>	<b>12</b>
3.1	Related Financial Paradigms . . . . .	12
3.1.1	Modern Portfolio Theory . . . . .	12
3.1.2	Options Theory . . . . .	13
3.2	Business-IT Alignment . . . . .	14
3.2.1	Business Process Management . . . . .	18
3.2.2	IT Governance . . . . .	19
3.2.2.1	Benchmarking . . . . .	20
3.2.2.2	Frameworks . . . . .	20
3.2.3	Service-Oriented Architecture . . . . .	21
3.2.4	IT Portfolio Management . . . . .	23
3.3	Valuation of IT Investments . . . . .	25
3.3.1	“Traditional” Cost Benefit Calculations . . . . .	25
3.3.2	Real Technology Options . . . . .	27
3.3.3	Traditional Multiobjective Optimisation . . . . .	27
3.3.3.1	Conventional Weighted-Formula Approach . . . . .	28
3.3.3.2	Lexicographical Approach . . . . .	28
3.3.4	Analytical Hierarchical Process . . . . .	28
3.3.5	Multiobjective Pareto Approach . . . . .	30
3.3.6	(Meta)Heuristic Methods . . . . .	33
3.3.7	Comparison of Valuation Methods . . . . .	33
3.3.8	Definition of Criteria Sets . . . . .	36
3.3.8.1	Critical Success Factors (CSF) . . . . .	37
3.3.8.2	Preferences of Stakeholders . . . . .	37
3.3.8.3	The OTSO Approach - Alignment to Business Strategy . . . . .	37
<b>4</b>	<b>Construction of the Software Selection Model</b>	<b>41</b>
4.1	Model Requirements . . . . .	41
4.1.1	Imperative Model Requirements . . . . .	41
4.1.2	Optional Model Requirements . . . . .	43
4.2	Concept . . . . .	44



4.2.1	Definition of Objectives . . . . .	45
4.2.2	Consideration of Time Periods . . . . .	49
4.2.3	Collection of Candidate Data . . . . .	49
4.2.4	Definition of Restrictions . . . . .	49
4.2.5	Definition of Dependencies . . . . .	50
4.2.6	Definition of Business Process Coverage . . . . .	52
4.3	Formalising the Model . . . . .	53
4.3.1	Creation of Portfolios . . . . .	53
4.3.2	Time Periods . . . . .	54
4.3.3	Benefit/Resource Categories . . . . .	54
4.3.4	Dependencies . . . . .	54
4.3.4.1	Inclusion/Exclusion Constraints . . . . .	54
4.3.4.2	Value-Modifying Dependencies . . . . .	55
4.3.5	Benefit/Resource Restrictions . . . . .	56
4.3.6	Business Process Coverage . . . . .	56
<b>5</b>	<b>Implementation</b>	<b>57</b>
5.1	Data Model . . . . .	57
5.2	Application Design . . . . .	58
5.2.1	User Interface . . . . .	58
5.2.1.1	Data Input Interface . . . . .	60
5.2.1.2	Evaluation Component . . . . .	60
5.2.2	Limitations of Implementation . . . . .	60
5.3	Stummer's INNOV Library . . . . .	61
5.3.1	Extension of Input Data Format . . . . .	61
5.3.2	Extension of Output Data Format . . . . .	62
5.4	Import of ADONIS Models . . . . .	63
<b>6</b>	<b>Comparative Case Study</b>	<b>66</b>
6.1	Input Decision Situation . . . . .	66
6.1.1	Base Characteristics . . . . .	66
6.1.2	Evaluation Criteria . . . . .	67
6.1.2.1	Application Requirements . . . . .	67
6.1.2.2	Design Specification . . . . .	69
6.1.2.3	Project Requirements . . . . .	69
6.1.2.4	Organisational Requirements . . . . .	70
6.1.2.5	Final Criteria Set . . . . .	70
6.1.3	Software Candidates . . . . .	71
6.1.4	Constraints . . . . .	72
6.2	Analytic Hierarchic Process . . . . .	73
6.2.1	Preconditions . . . . .	74
6.2.2	Calculating Objective Weights . . . . .	74
6.2.3	Ranking Candidates . . . . .	75
6.2.4	Composing a Portfolio . . . . .	76
6.3	Weighted Scoring Method . . . . .	78
6.3.1	Preconditions . . . . .	78
6.3.2	Ranking Candidates . . . . .	78
6.3.3	Composing a Portfolio . . . . .	80

---

6.4	Multiobjective Portfolio Approach . . . . .	80
6.4.1	Collection of Candidate Data . . . . .	81
6.4.2	Modelling Constraints . . . . .	81
6.4.3	Evaluation . . . . .	82
6.5	Comparison of Results . . . . .	85
6.5.1	Significance . . . . .	86
6.5.2	Analysis of Results . . . . .	86
6.5.3	Method Characteristics and Behaviour . . . . .	87
<b>7</b>	<b>Conclusions</b>	<b>90</b>
<b>8</b>	<b>Directions for Further Research</b>	<b>91</b>
8.1	Software Selection Model . . . . .	91
8.2	Implementation . . . . .	92
<b>A</b>	<b>Candidate Characteristics</b>	<b>93</b>
<b>B</b>	<b>AHP Comparison Matrices</b>	<b>96</b>
<b>C</b>	<b>MultiSel Program Masks</b>	<b>103</b>
	<b>List of Figures</b>	<b>109</b>
	<b>List of Tables</b>	<b>111</b>
	<b>Bibliography</b>	<b>113</b>

## Chapter 1

# Abstract

Because in the last decade, a strong coherence between business and IT as well as a thorough commitment to IT have become an important factor of competition on all market places and in nearly all industries, companies spent enormous amounts of money in information technology infrastructure [33]. Because of this, in the last few years many companies started to question their IT's *real* value and the total of benefits it generates. This usually induces a problem since modern software exposes many intangible characteristics that cannot be measured on common scales (such as usability, reliability or security). By that, valuation of software frequently becomes a complex, in some cases intractable task. Many modern IT management and aligning methodologies (such as IT portfolio management) emphasise a holistic management of IT landscape, which of course includes valuation of all employed software.

Today, there are multiple structured software selection approaches that enable evaluation of software. Though most of these methods - such as the Analytical Hierarchic Process (AHP) or Weighted Scoring Method (WSM) - permit finding the “best” out of a certain amount of candidates, they suffer from major disadvantages which render them unusable for repeated and sustained valuation of corporate software portfolios. Current methods for software evaluation employ the technique of preference weighting thus requiring an a-priori definition of objective rankings. This imposes the problem of decision makers having to formulate all restrictions prior to calculating candidate weights, which can result in the algorithms' having a low repeatability and their outputs' being biased to an uncertain extent. In addition to this, quality of solutions directly depends on the quality of a-priori data which makes the process of input data collection extremely sensitive. Another major drawback of current algorithms is that they are limited to evaluation of one single candidate at a time: Because most companies employ multiple software packages simultaneously, algorithms are demanded to consider groups (portfolios) of candidates together with their mutual dependencies.

Utilising the paradigms of portfolio theory and multiobjective optimisation, this thesis aims at developing a powerful yet straight-forward portfolio-aware software selection model that permits automatic, software-based evaluation of all possible candidate combinations under alignment to both corporate-level and project-level constraints. Multiobjective approaches do not aggregate criteria of different types into over-all values, nor do they require a-priori induction of preferences (e.g. objective weights). Thus, the new model respects the candidates'

real objective values and permits dynamic definition of preferences a-posteriori. Because no extensive collection of a-priori data is necessary, the approach yields high reusability permitting evaluation of whole corporate software portfolios on a regular basis with little effort and high transparency. This significantly improves manageability of corporate IT, reducing costs and leveraging efficiency of corporate business.

To test the new model's functionality, a decision support application is implemented to solve a real-world decision situation. Results are compared to those of two popular software valuation approaches, AHP and WSM.

## Chapter 2

# Introduction

In the last few years, many companies have not only changed their business scope and faced new opportunities but also consolidated their infrastructure because of the many possibilities that result from mature IT strategies [63]. The resulting increase in demand for professional IT services can even be visualised by the huge increase of IT service revenues of major IT business solution providers. Figure 2.1 shows how the product portfolio of IBM transformed between the years 1983 and 1997. While in 1983 a share of only 2 percent of IBM’s total revenue was generated with business services, it grew to 25 percent until 1997 [83].

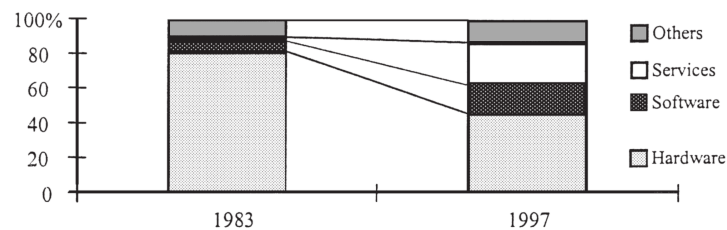


Figure 2.1: Evolution of IBM Service Revenues [83]

As the introduction of edge-of-technology IT was considered an important advantage in market competition, many large companies spent huge amounts for IT infrastructure expecting major benefits in overall business performance from an extensive integration of IT. Figure 2.2 shows the significantly increasing IT expenditure of Western European banks. From 1999 to 2004, total IT investments nearly doubled [33].

## 2.1 Motivation

Because of the enormous investments in information technology, companies started to question their IT’s *real* value and the total of benefits it generated. In a 2002 survey among 400+ top IT executives 60% reported an increase in the level of pressure to prove ROI on IT investments while 70% believe that their metrics do not fully capture the value of IT. Nearly half lack confidence in their ability to accurately calculate ROI on IT investments [82]. In 2004, Tiernan and Peppard noted that “despite all the investments made in IT, there is still considerable disappointment among executives with the return that has been achieved”.

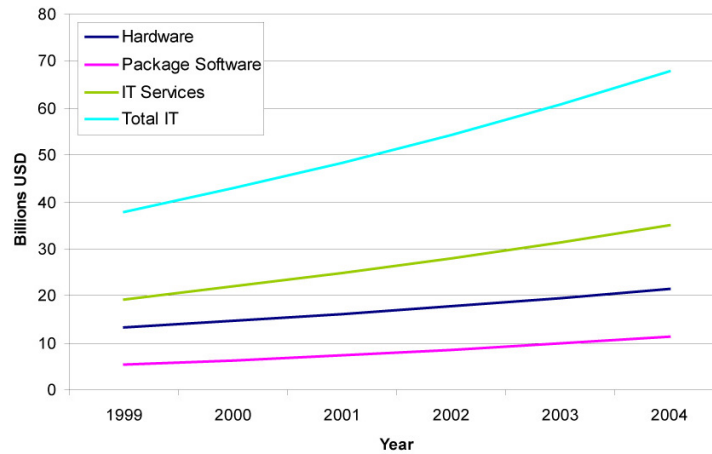


Figure 2.2: IT Expenditure of Western European Banks 1999-2004 [33]

They also stated evidence suggested that, though acquiring massive quantities of information technology, most companies did not possess methods to determine whether they had benefited from their investments at all [80].

Today, a strong coherence between business and IT as well as a thorough commitment to IT have become an important factor of competition on all market places and in nearly all industries [33]. Nevertheless, many companies still are unable to establish pragmatic and conscious management of their IT resources. In a 2003 study conducted by the Kellogg School of Management, 51 percent of the 130 questioned large companies reported that they had no process to evaluate IT investments against business strategy [7].

### 2.1.1 IT Management Approaches

To help companies catching up with this information deficit and to rendering both investing in and management of corporate IT more transparent, a lot of research has been done (cf. [44], [25], [51], [58]) and various IT management methodologies have been proposed. Whilst some of these approaches (such as IT governance) qualify as general guides for definition of corporate strategy under respect of information technological factors, others introduce frameworks that define concrete methods on how IT can be managed and integrated into corporate business. The common goal of all of these approaches is leveraging efficiency of IT and optimising IT spendings by aligning business and IT.

### 2.1.2 Valuation of IT Resources

Adequate valuation of IT resources (such as hardware, software, services etc.) is an important measure for evaluating costs, benefits and efficiency of a company's IT infrastructure. Though most business-IT alignment approaches recognise this requirement, none of them suggest concrete methods for determining the value of IT resources. The IT governance framework *COBIT*, for example, is limited to recognising the need of establishing "fair, transparent, repeatable and comparable evaluation of business cases including financial worth, the risk of not delivering a

capability and the risk of not realising the expected benefits” [46].

As a very important business object, a company’s software is very sensitive to select as it is an integral component of daily business. Of course, other IT infrastructural components such as IT hardware are important too, but because there usually is a big number of different hardware configurations that can support business applications, selection of the “right” (most efficient, most business-aligned, cost-optimal) software turns out to be a critical success factor.

Though this obviously raises the demand for well-defined and structured software valuation approaches, and though a considerable amount of work has been done on this topic (cf. [72], [38], [40], [57]), there are very few standardised and commonly accepted approaches for software valuation.

Because modern software exposes many intangible characteristics that cannot be measured on common scales (such as usability, reliability or security) but are important for overall software usability, valuation of software frequently becomes a complex and exhausting task.

This situation is even deteriorated by the fact that modern companies - in analogy to the paradigm of IT portfolio management - aim at analysing their business applications as a whole instead of valuing each application separately. By that, software valuation methods are demanded that consider much more than isolated characteristics of single software products, but characteristics of whole corporate software portfolios, inter-application dependencies, strategic requirements etc.

As the currently most promising approach for analysis of multi-application multi-characteristic decision situations, the multiobjective portfolio approach has already shown quality results with R&D project portfolio selection problems (cf. [76]). As pointed out by Neubauer [57], the method of multiobjective decision support (MODS) bears the capability of breaking down the complex process of software portfolio analysis to a step-wise, interactive approximation. At the same time, MODS permits consideration of characteristics of various types without having to “mix them up” by a-priori scoring or weighting.

## 2.2 Research Goals

The methodology of multiobjective decision support (MODS) yields several advantages over conventional objective weighting approaches (see section 3.3.5 for details). In the context of software selection, the most important of them are:

- No aggregation of candidate characteristics of different types, and by that no definition of objective weights is required.
- With its capability of analysing not only single candidates but sets of multiple candidates, MODS enables optimisation of whole portfolios of candidates to be used simultaneously. Mutual dependencies among candidates are considered as well as hard limits for the portfolios’ characteristics.
- As preferences are not induced a-priori but a-posteriori, results can be adjusted dynamically *after* the process of portfolio composition has been performed.

- The task of finding non-dominated portfolios is performed automatically, most of collected input data can be reused.

Because the MODS approach appears to deliver outstanding results in drilling down decision problems with many alternatives, and because the extension of the MODS method for use as software selection model seems reasonable, the main research questions of this thesis are formulated as follows:

- Which methods are currently used to evaluate and compare IT applications, and how effective are they?
- How can a generic MODS model be extended to enable efficient selection of software portfolios?

In course of this thesis, a fully functional MODS model for selection of software<sup>1</sup> portfolios will be developed. To prove the model's applicability, a proof-of-concept decision support software-tool will be implemented. After this, functionality of both model and application will be tested in a case study: A realistic decision situation will be analysed with the new MODS approach and results will be compared to those of other commonly accepted software selection methods.

The new MODS software selection model is supposed to significantly improve the process of software evaluation for organisations of various domains. Because proper management of IT implies a continuous monitoring of resources, companies should not only be given a tool to estimate the value of upcoming investments. They should also be enabled to evaluate their software portfolio in regular intervals, tracking its ROI by measuring costs and benefits thus optimising IT spendings and efficiency.

---

<sup>1</sup>The term "software" in this context applies to off-the-shelf components such as standard applications, (web) services etc.



## Chapter 3

# Related Work / Fundamentals

In the field of managing corporate IT, many research has been done. An overview over both the most common IT management approaches and the most popular existing software selection methods will be given in the following chapter.

## 3.1 Related Financial Paradigms

Financial paradigms such as portfolio theory, financial game theories or options theory have been adapted to other domains frequently. The following section introduces specific financial paradigms that have been adapted earlier for corporate IT management and IT investment planning.

### 3.1.1 Modern Portfolio Theory

Modern portfolio theory was established in 1952 when Harry M. Markowitz published his article “Portfolio Selection” in the Journal of Finance [48]. According to Markowitz [49], theory of financial portfolios differs from the classic economic theory of firm and customer in three major ways:

- The theory is concerned with investors rather than manufacturers and consumers
- Agents within this theory act under uncertainty
- The theory can be used to *direct practice*

The theory of the producer assumes that competitive firms know the price at which they can sell goods they produce. In reality, there is a delay between the decision to produce a good, the time of production and the time of selling it. Because of this, the final price of a good at selling time cannot be determined definitely at the time of making the decision to produce it. Due to this uncertainty, the decision whether or not to produce a certain product has to be made with a significant lack of information [49].

In the analysis of investor behaviour, uncertainty is an important factor. A fully-informed investor who exactly knew about the future returns of all stocks would surely choose nothing but the one stock that guarantees an optimal revenue. If there were multiple stocks with equal future revenues, the investor would be

indifferent among them, and there would be no reason to diversify the stock portfolio. So, as the reduction of uncertainty is the sole reason for diversified investment practices, it is essential in analysing rational investor behaviour [49].

Investment decisions do not only depend on the expectation of the maximum revenue, but also on the risk that is carried by each possible investment. Expected future returns can be calculated for each investment using formulas that incorporate time and possible revenue as well as risks represented by statistical distributions (e.g. mean and variance for a normal distribution) of final outcomes. Thus, risk carried by an investment is compensated to a certain extent by its higher potential revenue. An investor seeking to optimise revenues can select an investment portfolio by choosing a point from the set of Pareto optimal expected return/variance-of-return combinations known as *efficient frontier* [49].

Though classic portfolio theory was developed in a financial context, it can be applied in many other fields such as project selection or IT management. For example, the set of different software products (e.g. IT applications) simultaneously utilised by a company to support and maintain business processes can be considered the company's software portfolio [66] (see *IT Portfolio Management*).

### 3.1.2 Options Theory

Options are a popular approach for maximising ROI of upcoming investments while reducing risk. Purchasing an option reduces investment risk, as the decision whether to invest or not can be postponed to a time when return of investment can be predicted more accurately.

In 1973, a method for appropriate pricing of financial options was introduced by Black and Scholes. Because of the uncertain nature of future investments, the aspect of uncertainty was added to the approach of discounted cash flow (DCF).

$$OV = Se^{-\delta t} * \{N(d_1)\} - Xe^{-rt} * \{N(d_2)\},$$

$$\text{where, } d_1 = \{\ln(S/X) + (r - \delta + \sigma^2/2)t\} / \sigma * \sqrt{t}$$

$$\text{and, } d_2 = d_1 - \sigma * \sqrt{t}$$

Where,

OV = Option Value

S = stock price,

X = exercise price,

$\delta$  = dividends,

r = risk-free rate,

$\sigma$  = uncertainty (standard deviation),

t = time to expiry, and

N(d) = cumulative normal distribution function.

Figure 3.1: The BSM equation [73]

Basic component of the Black/Scholes/Merton (BSM) options valuation is the equation in figure 3.1. It allows calculation of option values from financial aspects (stock value, exercise price etc.) and risk parameters (risk-free rate and uncertainty) [73].

### Real Options

The term of *real* options came up as management consultants started to adopt the Black/Scholes/Merton (BSM) principle to deliver results regarding estimation of “real” options such as R&D project investments or identification of a company’s strategic directions [73]. Treating investment possibilities as real options yields multiple advantages, as the decision process is focused on calculating expected revenues under respect of inherent uncertainties (risks), explicitly accounting for the value of future flexibility [5]. Many decision processes and business activities can be cast to real option valuation (ROV) problems and valued using the BSM concept [73].

In 1997, McKinsey consultants Leslie and Michaels attempted to adapt the BSM concept to real options. They identified six levers in the BSM equation and matched them with their real options counterparts [41]. Similarities between financial and real options are illustrated in figure 3.2.

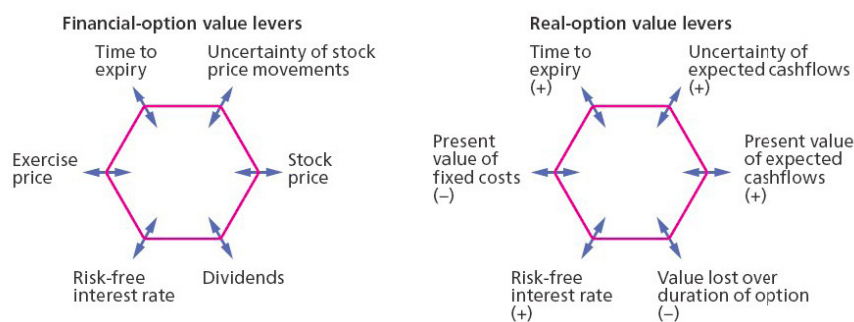


Figure 3.2: The six levers of financial and real options [41]

Effectivity of quantitative models is always limited by quality and precision of applied input parameters. Thus, many companies have been reported to face serious problems with adequately determining input parameters for real options valuation [5]. Circumventing this problem by creating more advanced and better-matching models is possible to a certain extent, but effectivity ceases as models become too complex. Significant behavioural differences between “traditional” real options and real options in IT (so-called real *technology* options) can be observed [53]. Details on real technology options will be given in section 3.3.2.

## 3.2 Business-IT Alignment

The rate at which new technology is introduced in modern companies is increasing by 20 to 30 percent every year [43]. To withstand competition, organisations are demanded to apply latest state-of-the-art technology strategically and accelerate innovation. But investing in IT blindly (as several companies did in the last decade) lacks effectivity as investments must be made precisely targeting a company’s business requirements. Morgan Stanley estimates that between 2000 and 2002, companies spent \$ 130 billions for IT they either did not need or could not use properly [51]. This, as well as the fact that pressure to prove ROI on IT investments is growing continuously [82], leads to a global demand for tools that help aligning IT expenditure to business.

In the last few years, many companies have not only changed their business scope and faced new opportunities but also consolidated their infrastructure because of the many possibilities that result from mature IT strategies [63]. A strong coherence between business and IT and a thorough commitment to IT have become an important factor of competition on all market places and in nearly all industries. Thus, aligning business processes with corporate strategy has become an important success factor. It enables development and implementation of cohesive organisational and IT strategies and allows companies to concentrate IT resources on value-generating and supplementary business processes. By understanding and patronising the partnership of business and IT, companies can optimise their IT spendings while leveraging efficiency of supporting and value-adding processes.

The “traditional” interpretation of business-IT alignment as “prioritising IT projects to support the priorities outlined by an organisation’s senior management” ([9]) experiences a much-expanded view. Today, there are many partly contrary concepts for business-IT alignment (see below). Luftman, for example, refers to business-IT alignment as *applying IT in an appropriate and timely way, in harmony with business strategies, goals and needs* [45].

Because many companies experience problems with adopting IT to match their business strategy, harnessing the power of information technology for their own long-term benefit, Luftman, Papp and Brier conducted a survey to determine most important *enablers* and *inhibitors* for business and IT strategy alignment. In interviews between 1992 and 1997, both business and IT executives were asked what they thought were the main enabling and inhibiting factors of business alignment. Weighting of enabling and inhibiting factors was uniformly distributed under business and IT executives. Only half of all questioned individuals believed that their companies’ business and IT strategies were aligned properly. In this survey under 500 organisations of 15 industries, the following top six enablers and inhibitors were determined [45].

Top six enablers (see figure 3.3):

- Senior executive support for IT
- IT involved in strategy development
- IT understands the business
- Business - IT partnership
- Well-prioritised IT projects
- IT demonstrates leadership

Top six inhibitors (see figure 3.4):

- IT/business lack close relationships
- IT does not prioritise well
- IT fails to meet its commitments

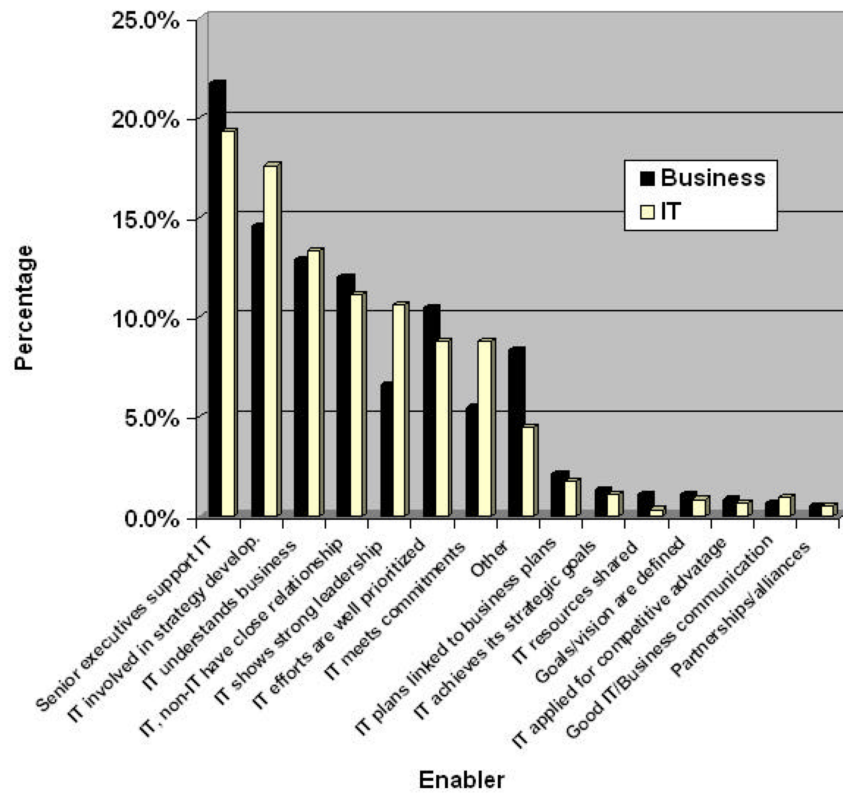


Figure 3.3: Enablers of business-IT alignment [45]

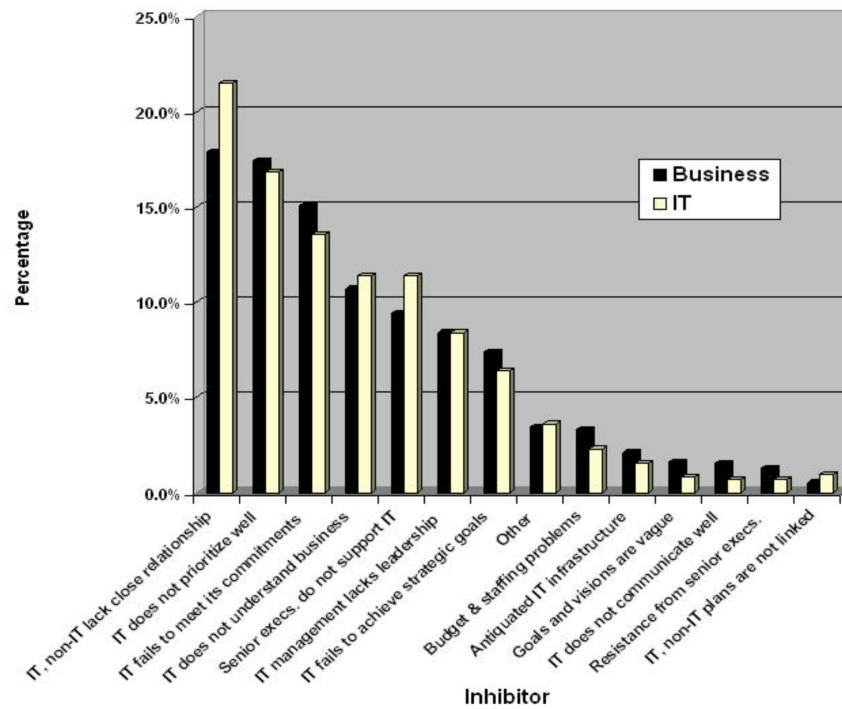


Figure 3.4: Inhibitors of business-IT alignment [45]

- IT does not understand business
- Senior executives do not support IT
- IT management lacks leadership

These results obviously endorse the popular and wide-spread assumption that the organisational management plays a key role in implementing business-IT alignment. To improve business-IT communication and leverage IT resources for building competitive advantages, IT should understand business and be involved in creation of enterprise strategies [45].

Reich and Benbasat [68] have created a model of business-IT alignment which accepts the social dimension as a central component. They raise the complaint that, in most of research done, no distinction is being made between the process of introduction of alignment and the final outcome, the state of business and IT being aligned. Thus, they differentiate between causal factors (such as the IS planning process) and the resulting state of alignment. They further divide the final outcome into two components, the *intellectual* and the *social* dimension [55]. The intellectual dimension refers to the alignment of IS plans (or strategies) with business plans (or strategies) and is defined as the state in which a set of high-quality interrelated business plans and IS plans exist [67]. The social dimension of alignment on the other hand refers to the state in which the IS and business executives understand and are committed to the business and IS mission, objectives and plans [68], [55].

Henderson and Venkatraman's *strategic alignment model* [24] explores the interrelationship between business and IT (see figure 3.5). It is focused on two distinct linkages, *strategic fit* and *functional integration*, and divided into 12 perspectives forming four quadrants (*business strategy*, *IT strategy*, *organisational infrastructure* and *IT infrastructure* and *processes* as well as *IT infrastructure* and *processes*). Interrelationships between perspectives (such as dependencies, interferences etc.) are represented by arrows.

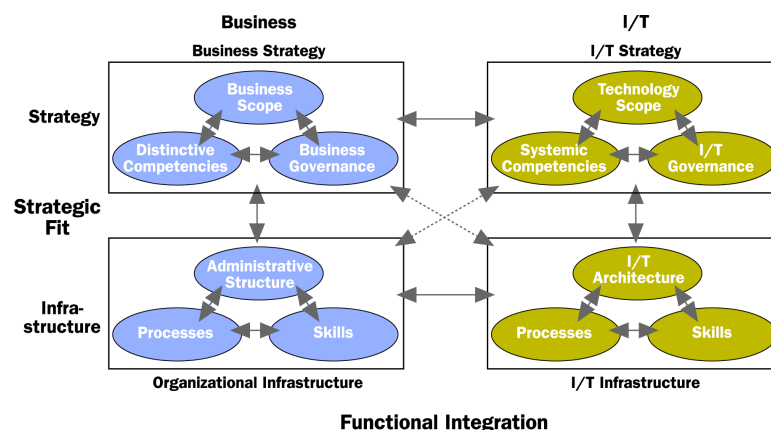


Figure 3.5: Strategic Alignment Model of Business and IT [63]

Once an organisation has defined its alignment perspectives, impact on financial performance must be evaluated. Therefore, measurements for the organisation's total performance (such as ROI, pre-tax income, net sales, growth etc.) must be found.

### 3.2.1 Business Process Management

Van Der Aalst defines business process management as “*Supporting business processes using methods, techniques, and software to design, enact, control, and analyse operational processes involving humans, organisations, applications, documents and other sources of information*” [2]. Since the late seventies, many differing methodologies for business process management have been proposed (see [32], [2], [17]). While in the early days, BPM was often performed with primary focus on software reengineering (cf. [23]), today’s it is thought of as continuous process to improve organisations’ business processes [58]. Thereby, BPM is limited to operational (“visible”) processes. All processes at strategic levels which cannot be made explicit are excluded.

Business processes are supported by IT resources. In the context of BPM, these IT resources usually are applications, services etc. that can be either components off-the-shelf (COTS) or custom-tailored components. Today’s companies’ increasing adoption of off-the-shelf component instead of using software developed in-house represents a major paradigm shift: In the seventies only few, mostly domain specific off-the-shelf components were offered. Thus, companies had to develop their own software solutions. Today, lots of general software as well as domain-specific applications are available as ready-to-use software packages [2]. Figure 3.6 displays the layers of applications and helps visualising the trends in information system development. In the sixties, the two inner layers were missing. Since then, following trends can be isolated:

- From programming to assembling,
- from data orientation to process orientation,
- from design to redesign and organic growth.

Regarding these shifts, Van Der Aalst correctly observes that “the challenge no longer is the coding of individual modules but orchestrating and gluing together pieces of software from each of the four layers.”

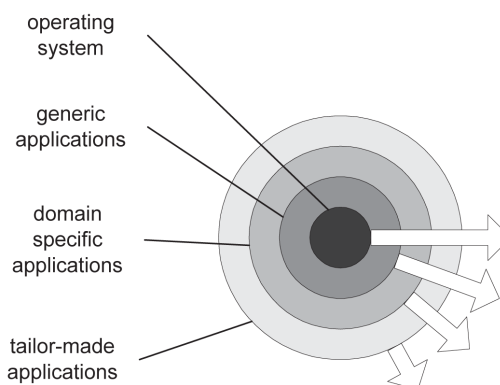


Figure 3.6: Trends in Information Systems [2]

When managing business processes (and by that, managing underlying components), valuation is a substantial task. Valuation should not only adhere to

explicitly expressed financial values like component's initial costs and ROI, but should also consider implicit and partly abstract properties such as extensibility, usability or performance.

Though a lot of work has been done regarding the paradigm of business process management, there is no common and standardised approach for valuing the processes that companies aim to manage thoroughly. Though several methodologies to close the gap between process management and valuation have been introduced (cf. [17], [8]), managers still tend to skip this partly complicated yet crucial task - often simply because of a lack of skills [58].

### 3.2.2 IT Governance

According to the IT Governance Institute, a well-structured and highly integrated adoption of supporting information technologies is important for optimising a company's IT spendings and maximising productivity [29]. Multiple approaches have been introduced to standardise efforts longing to reach this goal (cf. [84], [30], [22]). Currently, one of the most popular and wide-spread of these efforts is the paradigm of IT Governance.

IT Governance comprises a set of basic rules and measures to assure that within a company, all business processes are supported by suited IT resources, all resources are used responsibly and all risks are monitored adequately [81]. According to the IT Governance Institute (ITGI), "the overall objectives of IT governance activities are to understand the issues and the strategic importance of IT, to ensure that the enterprise can sustain its operations and to ascertain that it can implement the strategies required to extend its activities into the future" [29].

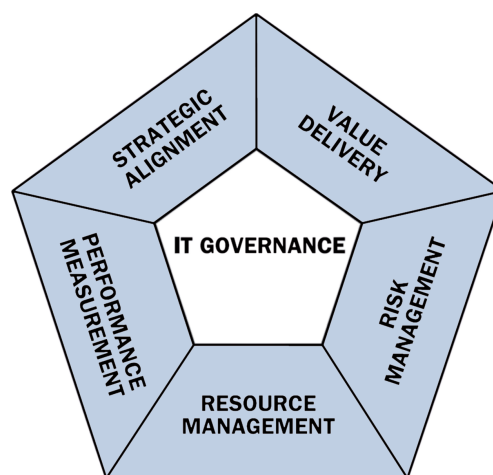


Figure 3.7: Aspects of IT Governance [46]

When introducing IT Governance in companies, several mistakes can be made. Forrester Research defines three elements crucial for the implementation of IT Governance [77]:

- **Structure**



Who makes decisions? Which organisational structures must be established?

- **Processes**

How are decisions made? Which processes are used to evaluate IT investments?

- **Communication**

How are decisions and process outcomes monitored, evaluated and communicated?

Especially the question regarding *processes* poses the problem of valuing IT investments and by that, valuing business software and its components.

### 3.2.2.1 Benchmarking

For determining and benchmarking the IT Governance maturity of organisations, the IT Governance Maturity Assessment is employed [21]. Just like the very general Capability Maturity Model (CMM) [64], the ITGMA contains classifications for 5 different maturity levels: Initial, repeatable, defined, managed and optimised. According to different studies, an average company implements IT Governance on level 2 (repeatable) [81].

### 3.2.2.2 Frameworks

There are different frameworks that facilitate implementation of IT Governance into organisations such as ITIL (Information Technology Infrastructure Library) [30] and COBIT (Control Objectives for Information and related Technology) [46].

ITIL was developed by the CCTA, an agency of the British government, in 1989. It essentially consists of a series of documents to aid with implementation of a framework for IT Services. ITIL comprises a set of best practices pointing out on what should be done, not how it should be achieved. The number of companies applying ITIL for IT service management is growing, companies that decided to adopt ITIL best practices can be certified (compliance to ISO 20000) [30].

The COBIT framework was developed by revisers of industry and the ISACA (Information Systems And Control Association) [81]. It relies on information gathered from business targets and required IT resources and processes and comprises 34 critical IT processes that are structured into the following domains: Plan and organise, acquire and implement, deliver and support as well as monitor and evaluate [46].

- for audit planning and audit program development,
- to validate current IT controls,
- to evaluate and reduce IT risks and
- as a framework for improving IT.

The number of organisations using the COBIT framework as well as the organisations' level and intensity of COBIT usage is increasing continuously. While COBIT 2.0 was used by most companies as a general guideline only, COBIT 3.0 is used by medium- and large-sized companies as a platform for control frameworks and audit processes [22].

Though both COBIT and ITIL emphasise importance of business-IT alignment and valuation of business processes, none of them provides suited tools for in-depth valuation of IT. COBIT, for example, just recognises the need of establishing “fair, transparent, repeatable and comparable evaluation of business cases including financial worth, the risk of not delivering a capability and the risk of not realising the expected benefits” [46].

### 3.2.3 Service-Oriented Architecture

Service-oriented architecture (SOA) aims at building independent, loosely coupled applications (*services*) that are created according to business process requirements. Every service provides a well-defined set of functionality that is abstracted from its origin and underlying implementation-related issues [36]. All services can be queried directly or combined with others to build new services, because they only depend on the definition of a standardised interface.

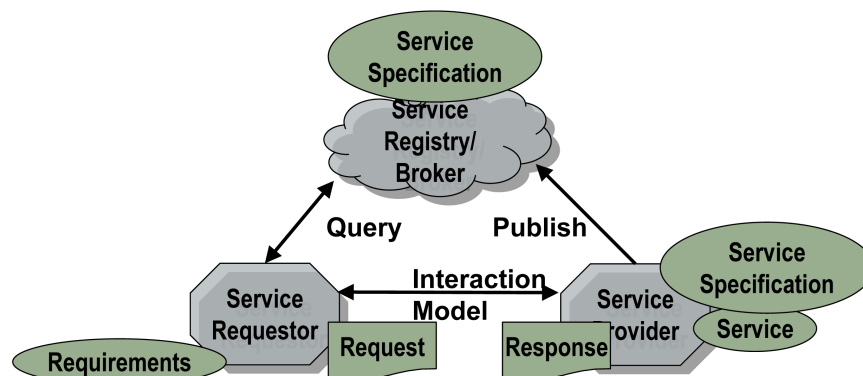


Figure 3.8: Service-oriented architecture - Find/Bind/Invoke [16]

Many large-scale ERP applications are static, unbudgeable monoliths. Extension and adaptation to new requirements very often is a tedious, difficult and by that expensive process. In contrast to this, and because of its modular nature, SOA yields the big advantages of flexibility and easy adaptation to new requirements. SOA can help to respond more quickly and cost-efficiently to changing market conditions while focusing IT expenses on business processes: As soon as business processes change, all underlying services are modified accordingly to reflect changes. If new services are required, they can be built upon existing ones (they can be *composed*) or implemented in any environment that supports the common service interface definition.

In a SOA, services are [16]:

- loosely coupled,

- directly usable,
- discoverable,
- accessible via standardised interfaces,
- context independent but
- not necessarily stateless.

To automate access to and interaction between services, a common registry of all available services is required. This so called *Find/Bind/Invoke* pattern is illustrated in figure 3.8. By querying a common registry, requesters obtain a reference to a service that can fulfil the required operation. After binding to the service, the requester invokes the desired operation.

The SOA framework outlines the different layers of SOA service interaction (see figure 3.9).

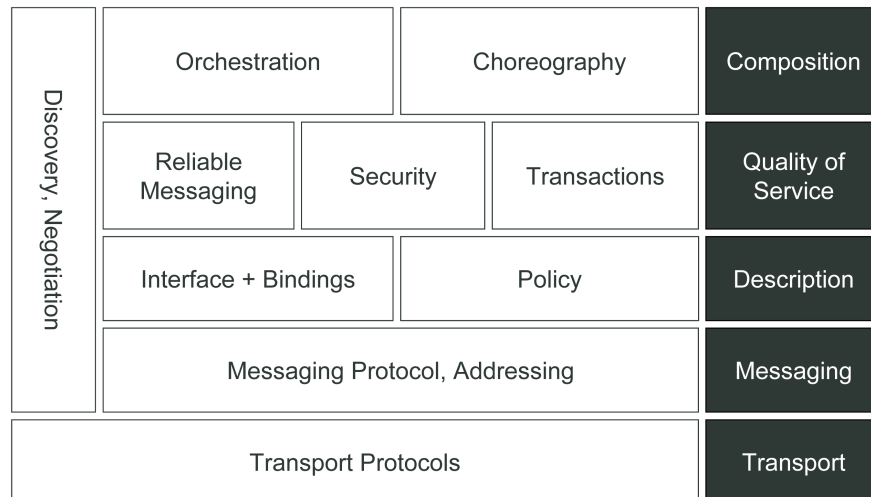


Figure 3.9: The SOA Framework [16]

As with selection of all components of IT, selection of services is important for aligning IT to business. Especially in context of loosely coupled and autonomous services, effective selection and combination of available technologies bears huge potentials of optimisation.

Currently, only few structured methods targeting automation of service selection and according decision support can be found. There are several papers coping with adaptation of services to changing environmental conditions (cf. [70], [27]), but most of them solely focus on the technical process of adapting services (or building easily-adaptable services) disregarding the task of refining optimal alignment of IT and business.

In analogy to multiobjective approaches coping with selection of R&D projects [76], Neubauer proposes a method that employs multiobjective analysis to provide decision support for selection of services in SOA environments [57].

### 3.2.4 IT Portfolio Management

In 1981, McFarlan introduced an approach to regard business IT resources similar to financial portfolios [52].

As initially mentioned, pressure to prove ROI on IT investments is increasing [82]. Managing an organisation's IT as a portfolio of services can help in achieving the necessary focus to deliver business value from IT investments [66]. Using this approach, decision-makers are forced to not only make their decisions on a short-term perspective, but to develop an over-all vision of IT investments. Thus, in 1996 the US Congress passed the *Clinger-Cohen Act* which compels government decision makers to adopt a portfolio approach to IT investments [33].

In analogy to financial portfolio theory (section 3.1.1), the central task in optimising IT portfolios is balancing them adequately. Especially IT portfolios can be tuned in many ways considering very different (and sometimes oppositional) objectives. Decision-makers have to assure their portfolios' balance in terms of risk, costs (ROI, TCO), technology, distribution, company strategy (alignment) etc. [33].

IT portfolio management is very often associated with managing portfolios of IT projects. Nevertheless, these IT projects are just a subset of object types within an IT environment that can be handled using a portfolio approach. On the other hand, IT portfolio management itself is one of multiple components that enable a holistic IT management in modern organisations. According to Kersten and Verhoef [33], the broad field of quantitative IT Portfolio Management consists in effect of four partly overlapping fields:

- **IT Portfolio Management**  
works with “Markowitz-like” models for IT, introduces balance to the IT portfolio, and works with IT portfolio assessment models and payback models.
- **IT Investment Management**  
takes a more investment-based approach, with the emphasis on return on investment (ROI), net present value (NPV), contribution to profit, substitution effects, etc.
- **IT Performance Management**  
deals with the assessment of the operational IT. Relevant subjects include IT dashboards, benchmarks, market conformity, the quantitative aspects of outsourcing, and service level agreements (SLAs), etc.
- **IT Due Diligence**  
focuses on the concrete quantification and realisation of synergy, reduction of the “time-to-harvest”, predictability and risk reduction of the integration, reduction of the “morning-after costs” etc.

Figure 3.10 shows a fictive IT project portfolio analysis. In this example, feasible projects are analysed in two dimensions: Alignment with company strategy and payback period (time-to-harvest). Projects that are hardly aligned with the company strategy or incorporate a relatively long payback period are considered

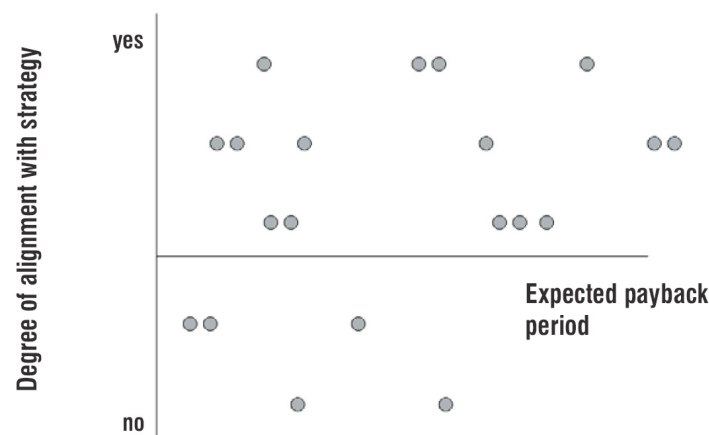


Figure 3.10: Example of a Simple IT Project Portfolio Analysis [33]

less eligible than those well-aligned to company strategy or with a short payback period. Starting with the most attractive projects, a project portfolio can be constructed until a certain constraint (desired number of project, maximum amount of initial spendings etc.) is met [33].

Five different kinds of IT portfolios must be distinguished [34]:

- **Hardware portfolios** contain all the hardware, infrastructure as well as interconnections among them. When no information about interconnections available, one can simply talk of an inventory account or investment account.
- **Service portfolios** seem reasonably easy to get for IT departments, as they relate to the operating and management activities of each department. Very often, Service Level Agreements (SLAs) give sufficient input for constructing correspondent portfolios. Practice however is intractable, mainly because it is always a matter of a whole chain of services or components, and shared functions or resource sharing (infrastructural services) are widely used.
- **Project and product portfolios** are the simplest portfolios. Techniques from the financial markets (like Markowitz' Modern Portfolio Theory) can be applied here easily with only minor adaptations. When managing such portfolios, statistical and other assumptions need to be given good attention.
- **Software portfolios** for this thesis are of primary interest. They can be partitioned in two independent sub domains, a) purchase strategies for ready-to-use software packages (sometimes referred as license portfolio management) and b) management of "classical" software development projects. Latter is one of the most challenging fields, since costs for software development are extremely high, diversity is huge and investment decisions are often very complex. Due to this fact software portfolio management bares an enormous potential of optimisation. Kersten and Verhoef "firmly" believe that "this is the main area in which successes can be achieved in the coming years with a good portfolio approach" [33].

### 3.3 Valuation of IT Investments

In a 2002 survey among 400+ top IT executives, Verhoef found out that “60% felt an increase in the level of pressure to prove ROI on IT investments while 70% believed that their metrics do not fully capture the value of IT” [82]. Nearly half of the questioned persons had no confidence in their ability to accurately calculate ROI on IT investments: “Most executives consider IT spending as a black hole: No matter how much resources are thrown at IT, there is no clear justification of returns - IT is on top of the executives.” [82]

Because of the continuously growing pressure to use IT efficiently, and as executives have learned that just *spending more* is not the winning strategy, methods for measuring total value of IT infrastructure are heavily demanded [72] [40] [45] [51] [67]. While total cost of ownership (TCO) can be determined relatively simple by summarising initial costs and subsequent maintenance costs, calculating the IT’s gain of efficiency as well as relative returns is difficult and highly approximative. In the following section, multiple popular models and methods that support quantitative IT measurement will be introduced. At the end of this section, methods with comparable pre- and postconditions will be juxtaposed to highlight each method’s specific advantages and drawbacks in the scope of software selection.

#### 3.3.1 “Traditional” Cost Benefit Calculations

The application of cost benefit calculation for selection IT applications is very common (cf. [8], [72]). Research on structured cost benefit analysis focusing information systems has been done since the late eighties. According to Peter G. Sassone [72], all cost benefit methods share some *basic principals*, some of which are

- money time value accounting under use of cash flow models,
- life cycle cost analysis to identify gamut of relevant costs,
- use of incremental (marginal) costs instead of average costs,
- recognition of net present value as the best financial criterion for aggregating costs and benefits over time.

Conventional methods for performing cost benefit analysis include [72]:

- **Decision Analysis**

This approach is an application of operation research techniques - namely Bayesian analysis and game theory - to business decisions. After the pursued objective is specified, a set of possible choices and states as well as the probability of states occurring and possible outcomes are defined. The value of an information system is expressed as the improvement of the objective value caused by the information system. With probabilities and pay-off represented in matrix form, the value of an information system can be visualised as the improvement of the expected value of the objective

subject to analysis caused by the use of the information system. Though this approach is useful for evaluating information systems that are used for routine decision making (e.g. credit decisions in a loan office), it is infrequently used because a) values of the probability-state matrices are difficult to determine and b) only few implementations of information systems fit into the decision analysis framework [35].

- **Structural Models**

This model analytically represents a company's business as a line of business functions. The value of an information system is estimated by monitoring the information system's impact on costs and revenues of the functions. As all of a company's relevant business operations must be modelled in mathematical formulas, complexity ranges from a few simple equations to very complex dynamic simulations. This approach directly links information systems' performance to the organisation's bottom line and helps with identifying relationships, parameters and assumptions. In addition to this, unrealistic expectations regarding the planned system are exposed. Unfortunately, this approach suffers from several drawbacks: It is difficult to use because very often links between information systems and the company's bottom line are hard to define. All models are unique, thus requiring a time consuming and costly development process. Usually, more data than available is required and details of the model become too complex to be communicated to the management. If the model is relative (measuring changes), there may be no usable audit trail.

- **Breakeven Analysis**

Breakeven analysis is a parametric assessment of benefits, where the parameter values are selected to equal costs and benefits. It is usually used when costs are quantifiable but key benefits are uncertain. Breakeven analysis can usually be performed quickly and easily, and is especially useful as long as benefits are extremely high or low (in comparison to alternate scenarios). If this is not the case, benefits have to be estimated independently.

- **Subjective Analysis.** Using this approach, decision makers are asked how much they would be willing to pay at maximum for the anticipated benefits of an investment (e.g. an information system). The result is compared to the costs of achieving the same goal with conventional measures. Because of its extreme subjectiveness, this approach is highly dependent on the quality of subjective estimation and by that on the knowledge and experience of the polled person. It is usually used with extreme caution and employed only when other (more objective) methods fail.

- **Cost Displacement/Avoidance**

The probably most common of these approaches compares the costs of a new information system to the costs of the system it displaces plus future costs it avoids. This approach assumes that the benefits of the new information systems are bigger or equal to those of the current information system, which cannot be considered true without further inquiry. This approach is conceptional straightforward, auditable, easily conveyable to management,

and appropriate data usually is simply to collect. Of course, this approach is of use only when there is a system to be replaced. As today more and more newly introduced information systems do not only replace knowledge workers but provide added value, the approach becomes increasingly inapplicable.

### 3.3.2 Real Technology Options

Theories of traditional real options (see section 3.1.2) and real technology options significantly differ in certain aspects as conventional valuation techniques show their limitations when being applied to investments that require major commitment under conditions of significant uncertainty [53]. Financial options models calculate an option's price under several core assumptions. Some of these are

- that an option's underlying asset must be priced,
- this price must be known,
- assets must be continuously tradeable and
- the value of an option increases with the volatility of its underlying asset.

For technology investments, this model is not fully valid. As an example, technology assets are not continuously tradeable, and prices of underlying assets are not always known [53]. Nevertheless, there are many parallels between financial and real technology options. The *price* of an option for example can be equalled to the costs of an initial software development. Further, such an option can be *exercised* by additional investments required for the software's commercialisation. Finally, as with market launch, another asset is created; the software brings returns and can be traded (sold, licensed, spun out etc.).

Dixit and Pindyck [14] point out that there are three different types of uncertainty in the context of real technology options: *Input cost uncertainty* refers to factors that are external to the company and cannot be influenced. *Technical uncertainty* is the uncertainty that arises from general risks of technology development projects. Examples are additional costs, unexpected delays or even the project's failing completely. Unlike input cost uncertainty, technical uncertainty can be influenced. The mostly-used way of reducing technical uncertainty is by investing. The third uncertainty is made up of factors that are located external to the company but can be influenced though, e.g. by strategic decisions. Depending on where an option's uncertainty is located, its value must be determined regarding the specific *boundary conditions* and strategic action must be taken accordingly.

### 3.3.3 Traditional Multiobjective Optimisation

When ranking candidates, their specific parameters are compared between each other. This can either be done using only one or using multiple characteristics of each candidate. The latter case usually poses the problem of having to consider



more than one criterion per candidate which renders a straight-forward arithmetic ranking of candidates without further examination impossible.

Multiobjective methodologies explicitly focus on comparing and ranking multiple selection candidates (or even sets of candidates) in respect of multiple objectives [76]. Multiobjective decision-making has been studied extensively over the past few decades [75] and by now, there are many different approaches applying multiobjective techniques to solve a broad bandwidth of decision problems [18]. While many more traditional approaches cope with multiple objectives using combinatoric techniques such as aggregation or weighting, other approaches (such as the multiobjective Pareto approach, see 3.3.5) use algorithms that treat objective values of different types on their own which yields many benefits (again, see 3.3.5). The most common traditional multiobjective approaches include [18]:

### 3.3.3.1 Conventional Weighted-Formula Approach

This - in literature by far mostly used approach (cf. [18]) - is a classical weighted scoring method (WSM). It aims at transforming multiobjective problems into “trivial” problems with only one single objective. By weighting each of the objectives numerically and combining the weighted criteria into a single numerical weight for each candidate, all candidates can be easily compared. The weighted-formula approach is one of the conceptually simplest methods that respect multiple objectives. Though it is fast to perform, this method bears significant drawbacks (see section 3.3.7).

### 3.3.3.2 Lexicographical Approach

The basic idea of this approach to assign different priorities to all objectives and compare all candidates under respect of their objectives’ priority. Candidates are compared to each other starting with the highest priority and descending until one candidate’s objective value exceeds the other one’s. After comparison, a clear ranking of all candidates is possible. Drawbacks of this method are the difficulty of a priori prioritising weights (expressing preferences) for all objectives and the systemic inaccuracy induced by aggregation of objectives of different types: When, for example, software products are evaluated, it is very complex to find a common scale for parameters like *costs*, *usability*, *scalability*, *maintainability* etc.<sup>1</sup>

### 3.3.4 Analytical Hierarchical Process

The Analytical Hierarchical Process was introduced by T. L. Saaty in 1980 [71]. It is a structured method for comparing quantifiable and intangible criteria of multiple candidates, and is commonly used to assist with multiobjective decision problems (cf. [17], [85], [40], [37]). There are also decision support frameworks that use the AHP as core component for candidate comparison (cf. [37]). The AHP is usually performed in three stages:

---

<sup>1</sup>See section 3.3.8 for more details on comparability of criteria sets.

- **Decomposition**

At the beginning of the AHP, a decision network is created. The specific decision situation is modelled as a hierarchic network with the top representing overall objectives and lower levels representing criteria, sub criteria and alternatives (see figure 3.11).

- **Comparative judgements**

Then, participants set up a comparison matrix that compares pairs of criteria or sub criteria of each hierarchy level. Weights are employed to express the users' preferences for one of the two compared criteria numerically: A rating of 9 means "total preference for criterion A over criterion B",  $\frac{1}{9}$  means "total preference for criterion B over criterion A", and 1 means "ambivalence between criterion A and criterion B" (see table 3.1).

- **Synthesis of priorities**

Finally, based on preferences derived from criteria comparison matrix, composite weights of all alternatives are calculated [40].

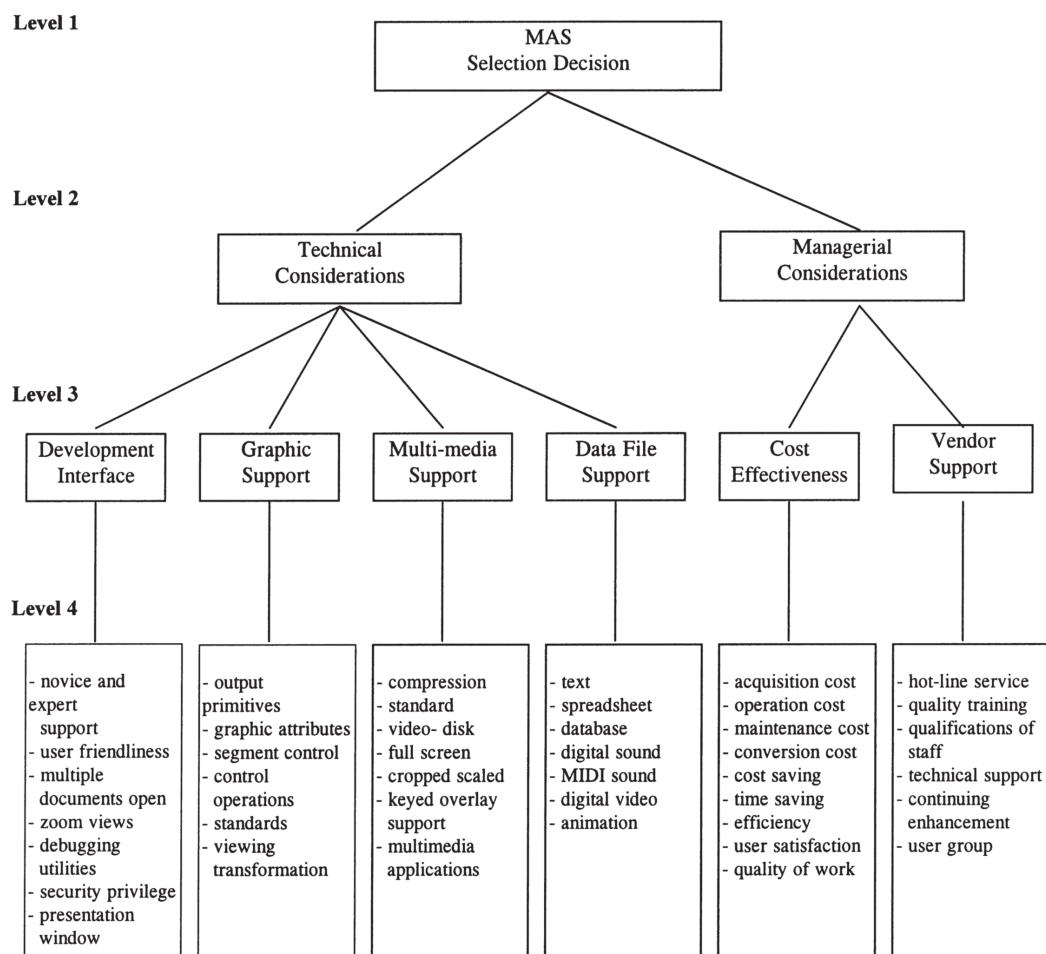


Figure 3.11: Example: AHP decision network [40]

The strength of the AHP method lies in its ability to structure complex, multi-person, multi-attribute, and multi-period problems hierarchically [85]. Unfortu-

CRITERIA	A	B	C	D	E	F
A. Dev. interface	<b>1</b>	1/3	1/4	1/3	5	7
B. Graphics support	3	<b>1</b>	1/3	1/4	5	6
C. Multi-media support	4	3	<b>1</b>	2	7	8
D. Data file support	3	4	1/2	<b>1</b>	6	8
E. Cost effectiveness	1/5	1/5	1/7	1/6	<b>1</b>	5
F. Vendor support	1/7	1/6	1/8	1/8	1/5	<b>1</b>

Table 3.1: Example: Criteria Comparison Matrix [40]

nately, there are multiple drawbacks when utilising the AHP for software selection: Weighting criteria against each other often induces major uncertainty as many different types of criteria cannot be directly compared (see section 3.3.8). In addition to this, criteria have to be weighted subjectively which can cause a significant bias. As the number of candidates (alternatives) and evaluated attributes increases, the required manual effort becomes very high.

### 3.3.5 Multiobjective Pareto Approach

The Pareto approach solves multiobjective decision problems using the paradigm of Pareto efficiency<sup>2</sup>. All possible Pareto efficient solutions<sup>3</sup> within solution space can be found by comparing all possible solutions and dropping those that are dominated by at least one other solution.

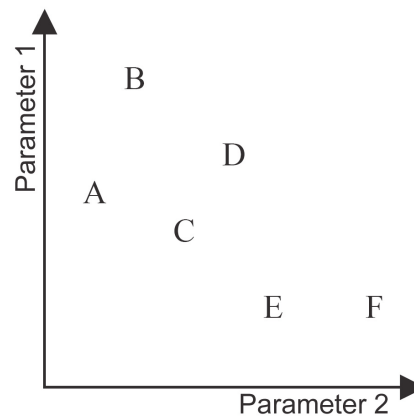


Figure 3.12: Example for Comparison of Portfolios with Two Parameters

Figure 3.12 shows a simple example for multiobjective comparison of solutions: Solution B, D and F are Pareto dominant in regard of the two compared parameters. All other solutions are dominated because they are superseded by other solutions in all (two) objectives (*Parameter 1* and *Parameter 2*).

<sup>2</sup>A solution  $s_1$  is said to dominate (in the Pareto sense) a solution  $s_2$  if and only if  $s_1$  is strictly better than  $s_2$  with respect to at least one of the criteria (objectives) being optimised and  $s_1$  is not worse than  $s_2$  with respect to all the criteria being optimised [18].

<sup>3</sup>A solution  $s$  is said to be Pareto efficient if it is not dominated by any other solution.

Using the Pareto approach yields big advantages: Without any a priori weighting of criteria, solution space is reduced until there are only solutions left that are “not obviously *worse* than other ones”. All criteria are compared to equivalent criteria, and there is no need to compare them to criteria of different type. The Pareto approach aims at significantly reducing solution space while keeping the quality of results constant and not inducing any uncertain or estimated variables. Because remaining solutions are not ranked in any way, additional screening methods are required to determine a single solution that fits best overall objectives. An overview of interactive solution screening methods was published by [19], Neubauer and Stummer presented an interactive screening method for COTS selection [59] and extended business process management methodologies for evaluation of IT-investments [58].

### **Stummer/Heidenberger MODS Approach**

Stummer and Heidenberger introduced a model that adapts the multiobjective Pareto approach (MODS) to permit selection of R&D portfolios<sup>4</sup> [76]. By that, not only single non-dominated project candidates but whole non-dominated project portfolios can be sought.

Many companies manage their research and development (R&D) projects as a portfolio. This method is becoming more and more popular as it allows decision makers to develop an over-all vision of IT investments. Since 1996, US government decision makers are compelled to adopt a portfolio approach for their IT investments [33]. Because selecting efficient project portfolios is very important for R&D companies, the S/H MODS approach can significantly improve R&D performance and optimise ROI [76]. The S/H method aims at finding the most attractive project portfolio for an organisation. This is accomplished by not only considering objective values of candidates (projects) within portfolios but also predefined baselines (such as minimum and maximum value for each objective) and specific interdependencies among candidates. Moreover, portfolios can be evaluated over multiple time periods to determine portfolios’ performance under varying conditions within a certain duration [76]. The approach essentially consists of the following four separate phases:

- **Screening procedure**

The initial screening procedure identifies projects that worthy further evaluation and thus helps keeping the number of projects entering in-depth analysis within manageable size. Screening is done using a scoring procedure, a method that has been applied to R&D project selection since the 1960s [31]. This procedure yields no results of high accuracy, but it allows narrowing down the number of possible evaluation candidates to a reasonable number without having to collect much data for each project and without discouraging participants. This phase should output not more than about 30 projects as computation increases significantly.

- **Definition of benefits and resources for each objective**

To provide information for subsequent comparison of projects, all candidates from the first phase are subject to further in-depth analysis under use

---

<sup>4</sup>See section 3.1.1 for more information regarding portfolios.

of a multi-round Delphi process. Each project's owner provides in-depth information that serves as input for evaluation. In a multi-stage Delphi process, participants discuss benefits and resource consumptions for each project as well as dependencies among them. For all candidates, a common set of evaluation criteria is derived (see section 3.3.8). General constraints regarding the composition of portfolios as well as interdependencies (inclusion or exclusion conditions, synergies etc.) among specific projects can be defined.

- **Determination of Pareto efficient portfolios**

Out of all possible combinations of all projects from phase two, this phase aims to isolate all Pareto efficient portfolios under respect to all the projects' objectives and all restrictions defined in phase two. This is accomplished by enumerating all possible combinations of candidates (constructing all possible project portfolios) and checking them both for Pareto efficiency and against defined restrictions. All portfolios that are dominated or do not obey predefined restrictions are dropped.

- **Restriction of solution space**

In the last phase, upper and lower boundaries are defined for each objective restricting solution space until only few solutions are left. This is accomplished using an interactive screening method ("solution space explorer application"). The application takes all remaining portfolios as input and visualises distribution of solutions graphically (see figure ). Finally, the few remaining solutions are subject to manual evaluation and discussion. The most attractive portfolio is chosen.

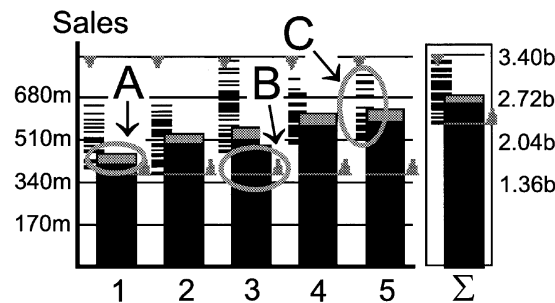


Figure 3.13: Restriction of solution space [76]

The S/H approach yields two major advantages in comparison to other multi-objective approaches:

- **A posteriori definition of preferences**

When stakeholders have to express their preferences at the beginning of a valuation process (see AHP), output cannot be refined easily without repeating the whole decision process. The S/H approach lets users define their preferences at the very end of the process. While reducing solution space, users can see the consequences of their actions (namely, the impact on the number of remaining portfolios) in real-time. Preferences can easily be adjusted until only a few solutions remain.

- **No aggregation of objectives**

Many approaches use criteria weights to aggregate all selection candidates' objectives to a total weight (again, see AHP). This confronts users with the problem of having to assign priorities to all evaluated objectives. Especially with software, this is quite pointless as many objectives very often cannot be weighted in respect to each other. Because of this, most users will make their choice based on intuition instead of logic, which induces a significant factor of randomness.

As this methodology yields good results with R&D project portfolios, and because the process of project portfolio selection shows many similarities to IT software portfolio selection, the Stummer/Heidenberger method will be used as basis of this thesis' model for software portfolio selection. For details on how this is done and which extensions are implemented, see chapter 4.

### 3.3.6 (Meta)Heuristic Methods

Very often, formal decision making approaches require notable computation efforts. The multiobjective portfolio selection technique, for example, requires a complete enumeration of all possible portfolios (namely all possible subsets of all candidates). While this can be performed within acceptable time for comparatively small problems, it becomes increasingly demanding as the number of projects grows because computation effort increases with  $2^n$  [15].

Heuristic paradigms offer significant reduction of computation expenses and time when applying problem solving algorithms. Their common major drawback is the loss of certainty that arises from their heuristic (and thus approximative) nature. Because of this, the challenge in applying heuristic algorithms consists of selecting and adjusting applied algorithms in a way that not only reduces computation effort but also preserved quality of results to a feasible extent. Today, there are many different heuristic approaches for decision and computation problems. For example, ant colony optimisation [15] or genetic algorithms [28] can be applied to increase efficiency of solving multiobjective decision problems. Because this thesis' primary aim consists of the creation of an appropriate model to perform software selection, no fine-tuning is performed and thus no heuristic methods are applied. Moreover, the number of software candidates will be restricted to a reasonable maximum. Further information regarding the application of heuristic methodologies on multiobjective optimisation can be found in [15] and [28].

### 3.3.7 Comparison of Valuation Methods

Comparing methods and algorithms of different characteristics turns out difficult as a common set of measurement criteria has to be found. Some of the mentioned approaches cannot be compared to others because they are either abstract models that must be used with a concrete method to yield results, or they are strongly focused on certain single valuation aspects what makes them inefficient to use for valuation of software (e.g. many traditional cost benefit calculation methods). All relevant of the remaining approaches are compared using a set of six criteria (see table 3.2).

*Note: All ratings have been made with substantial focus on the task of software selection. In other contexts, approaches might perform considerably better or worse. For each valuation approach, usually there are multiple differing interpretations and implementations. The models and algorithms compared here have been taken from their respective literature references. For some algorithms, there might exist different forms and models that might have been rated differently in this comparison.*

The following criteria have been chosen in regard to their relevancy for the domain of software selection:

- **Multiple objectives**

When evaluating software, consideration of multiple objectives is of high importance, as a software's adequacy and efficiency for a company is determined by many different parameters [40]. Parameters of different characteristics which cannot be measured on a single common scale must be compared among software candidates. Therefore, expression of parameters in monetary values only is no adequate solution as many parameters of software simply cannot be cast.

- **Difficulty**

The difficulty of valuation approaches is an important criterion either. The more difficult an approach, the more unusable the whole model becomes to the users. An ideal approach should yield good results while keeping complexity<sup>5</sup> (and by that difficulty) at a minimum. Difficulty of an approach can only be measured approximatively. For example, one could measure the time it takes the users to "learn" a method, or the degree of knowledge required for the users to understand an approach.

- **Required "manual" interaction**

This corresponds to the degree of manual effort an approach requires. The more manual tasks have to be performed, the less the approach is repeatable to the users and the more uncertainties are introduced. Required manual effort could, for example, be measured by the time it takes to perform evaluation of a previously defined set of reference candidates.

- **Applicability to portfolios**

When evaluating software, it is important to not only consider each candidate separately but combinations of candidates. Similar parameters of portfolio candidates cannot always be arithmetically summarised to get the portfolio's overall performance, and dependencies between candidates must be considered. Thus, approaches must include suited methods to compare portfolios.

- **Repeatability/reusability**

Holistic approaches like business-IT alignment do not only imply one-time evaluations of software but rather continuous evaluation of the current

---

<sup>5</sup>Note that in this context, the term *complexity* does not imperatively correspond to the concepts of mathematical or computation complexity rather than the difficulty of performing the method for the user(s).

software’s overall performance. When external preconditions or certain software candidates’ characteristics change, re-evaluation of all candidates should be possible easily and straight-forward.

Repeatability of an approach in terms of required effort for each subsequent repetition can be measured by comparison of required initial  $e_i$  to subsequent  $e_s$  effort. The smaller the ratio of subsequent to initial effort, the more repeatable an approach (see equation 3.1).

$$R_{max} : r = \frac{e_s}{e_i} \quad r \rightarrow 0 \quad (3.1)$$

- **Quality of results**

The best approach cannot be useful if quality of its results is poor. Nevertheless, a certain level of weakness in result quality might be acceptable if overall results of the respective approach are very outstanding and weaknesses regarding results (e.g. variance of results) are well-known.

There are multiple methods for determining the quality of results. They can, for example, be compared to measured real-life values or be validated by simulations (see [12]). If a method incorporates aspects of uncertainty (i.e. random variables), a certain indicator for the quality of results can be derived from the result’s intrinsic variance.

APPROACH	MO	C	RMI	AP	R/R	QoR
1 Breakeven Analysis	No	Medium	Medium	Yes	Low	Low
2 Decision Analysis	Partly	High	High	Yes	Low	Medium
3 Real Tech Options	No	Medium	Medium	Yes	Medium	Medium
4 Weighted Formula	Yes	Low	High	Yes	Low	Low
5 AHP	Yes	Medium	High	No	Medium	Medium
6 Pareto Approach	Yes	Medium	Medium	No	High	High
7 S/H (MODS) Approach	Yes	Medium	Medium	Yes	High	High

Table 3.2: Comparison of applicable valuation methods [40]

Table 3.2 visualises the comparison of valuation methods. While more “traditional” approaches aggregate values by plain calculation (e.g. breakeven analysis, decision analysis), current software valuation methods aim at structuring decision situations and applying principles known from other domains such as economy (e.g. options theory, Pareto paradigm). Each of the listed methods has individual advantages and suffers from specific drawbacks. In general it can be stated that, the more complex an algorithm, the harder it is to apply and the more error-prone it is. The more a-priori parameters of different types are aggregated, the more bias will appear in the final result.

Currently, Pareto approach and MODS approach are the only ones that are designed for considering combinations of candidates. While all other approaches combine candidate characteristics, only the MODS approach treats candidate characteristics separately and thus yields results without requiring a-priori weighting. While certain methods, such as the AHP or MODS, provide precise



though generic decision models that can be used mostly without adaptation to the specific decision situation, other approaches like decision analysis require putting up very custom and specific models.

Several models have been proposed to automate, optimise and - above all - simplify the process of software selection. While traditional approaches of cost/benefit calculation can usually be applied to software selection decision problems under major limitations only (see section 3.3.7), some recent approaches based on either the AHP or multiobjective optimisation tend to cover software characteristics on a higher level delivering reasonable results (cf. [85], [40], [76], [58]). Though, as analysis of whole software portfolios yields several additional challenges and problems, even approaches like the AHP appear to be applicable to a limited extent only.

### 3.3.8 Definition of Criteria Sets

Whenever objects are compared, ranked or selected, the final outcome always strongly depends on the choice of which characteristics to consider and measure. Optimal results can be obtained only if the *right* (for the specific decision situation best-fitting) set of criteria is chosen.

In 1979, Boehm et al. defined a set of software evaluation criteria that was supposed to enable a quantitative evaluation of software [4]:

- Portability
- Reliability
- Efficiency
- Human engineering
- Testability
- Understandability
- Modifiability

Though this list of criteria seems comprehensible and convincing, today it does not only lack actuality but also specificity to particular application domains. In terms of business-IT alignment, software that is aligned with corporate business can only be found if evaluation criteria that lead to the selection of one or multiple candidates are aligned to business either. Thus, criteria must be not only aligned to business processes but to corporate business strategy as a whole in order to achieve an optimal degree of business-IT alignment [9]. Though specification of suited criteria sets obviously is a substantial factor for good results, only few structured, repeatable and efficient approaches for determining software criteria sets have been proposed.

### 3.3.8.1 Critical Success Factors (CSF)

The method of critical success factors was developed by *McKinsey & Company* [10] in 1950. CSF is a methodology that systematically identifies those actions that are necessary to enable an enterprise to achieve its goals [69]. The CSF method identifies all areas of activity in which good result are absolutely necessary for sustaining business. From these activities, a) requirements can be derived or b) they can directly be used as input for approaches like the AHP [17].

### 3.3.8.2 Preferences of Stakeholders

Multiple approaches for software selection accomplish the necessary task of criteria selection under consideration of the involved stakeholders' preferences ( [58], [56], [85], [76]). This can be done using several methods:

- **Interviews**

Decision makers and project stakeholders can be interviewed to gain information regarding important criteria [50] [85].

- **Delphi processes**

The Delphi method<sup>6</sup> permits forming of common opinions by discussion while mitigating influence of dominant individuals [76].

- **Workshops**

Moderated workshops can be used as structured methods for forming common opinions among multiple persons (cf. [56], [58], [76]). Within a workshop, specialised methods (such as Delphi processes) can be employed.

### 3.3.8.3 The OTSO Approach - Alignment to Business Strategy

Because the primary goal of software employed in a company is supporting the company's business processes (thus helping to pursue business strategy), criteria for selecting software must be aligned to business strategy as well in order to optimise the software's overall business coverage. OTSO aims at accomplishing business strategy alignment by deriving criteria from a basic set of factors aligned to business.

With *OTSO (Off-The-Shelf Option)*, a selection framework for reusable software components, Kontio describes a way of criteria definition that strongly adheres to the company's business strategy [37]. Though OTSO lays its focus on reuse of components, the model bears two general approaches that appear very useful for selection of many types of software components:

- **Incremental criteria development**

The OTSO model defines selection criteria in an incremental and evolutionary process. Criteria can be gradually refined to ensure an optimal matching of influencing factors and software selection criteria.

---

<sup>6</sup>See [42] for details on Delphi processes

- **Hierarchical criteria derivation**

Figure 3.14 shows the hierarchic dependencies that are used for derivation of criteria. Starting from the most important influencing factors such as organisation infrastructure, application architecture or project constraints, information is collected and requirements are broken down until specific selection criteria can be deduced.

The OTSO model recognises five main factors (given preconditions and requirements) that collection of software selection criteria depends on:

- **Application requirements**

As the most important of these factors, application requirements can be divided into two categories, functional requirements (e.g. coverage of specific business processes) and non-functional requirements (such as performance or maintainability). If available, requirement specification should be used for interpreting such requirements.

- **Application architecture and design**

In this context, application architecture and design determine how systems are built. This not only includes components and design patterns but also interfaces and communication standards.

- **Project objectives, constraints**

Selection of components can be influenced by project constraints such as time or budget. For example, if introduction of a new software depends on specific budget constraints, this must be regarded when selecting among possible alternatives.

- **Organisation infrastructure**

Components must be selected in respect to and under consideration of the organisation's infrastructure.

- **Availability of libraries**

Software libraries are an important factor for reusability of software. On one hand, an organisation's existing libraries can be reused which may leverage costs. On the other hand, new software components that include reusable libraries can save resources on future software acquisitions.

This factor is of less importance for generic software selection models as the criterion of reusability usually is considered just as important as others. If required, criteria regarding reusability can also be derived from the factor of application architecture and design.

Kontio also identifies four classes of evaluation criteria:

- **Functional criteria**

Criteria that correspond to the components' functional aspects and capabilities, such as supported graphic formats (e.g. imaging applications), maximum of session (e.g. database applications) etc.

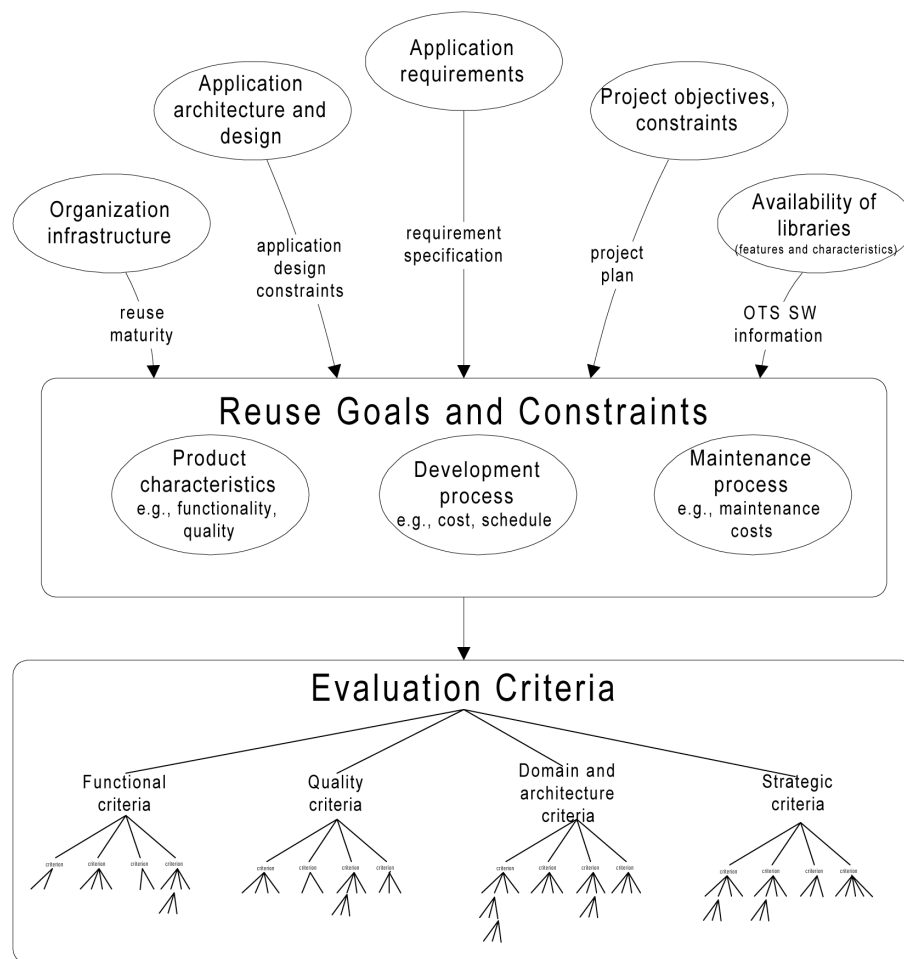


Figure 3.14: Factors influencing selection of criteria [39]

- **Quality criteria**

Criteria that correspond to software quality aspects, such as performance, stability, usability etc.

- **Domain and architectural criteria**

Criteria that correspond to requirements specific to the organisational domain and architectural aspects, such as compatibility with present systems and interfaces etc.

- **Strategic criteria**

Criteria that correspond to business strategy and strategic project constraints, such as initial costs, life-time cycle etc.

The OTSO model appears suitable as a basis for construction of a best possible set of software evaluation criteria. As the model subject to this thesis does not primarily focus reusability of libraries but considers reusability a criterion as many others, certain changes will be induced in the OTSO model that render the process of criteria set definition more generic and open to evaluation of various types of software (such as components, services, libraries etc.).

## Chapter 4

# Construction of the Software Selection Model

None of the currently existing methodologies for software selection (or methods for software valuation that enable selection of software by comparison to other candidates) fully satisfy the needs of today's holistic IT management paradigms (such as business-IT alignment): While many paradigms lack the ease of use required for performing business application portfolio evaluation on a regular basis, others are limited to evaluate of only very few candidates. Several approaches are not capable of coping with portfolios of applications whilst others do not align software selection to business processes' demands. In this chapter, a model will be developed that bypasses the shortcomings of other popular software selection models.

## 4.1 Model Requirements

The following sections list requirements a software selection model needs to cover in order to permit efficient and versatile evaluation of software portfolios. A differentiate is made between *imperative requirements* that are substantial components of the model and *optional requirements* that will be interpreted as general objectives which should be met.

### 4.1.1 Imperative Model Requirements

Characteristics of and dependencies among business software components are various and partly quite complex. Some components, for example, may require others to function properly (e.g. certain software libraries), while others can hardly be applied together with certain other products (e.g. *Microsoft Internet Information Server (IIS)* and the *Apache Web Server*). When using some products simultaneously, they perform better or worse than they would alone.

Some applications bear higher initial costs and low maintenance costs, others can be obtained totally free of charge (e.g. open source software) but cause high morning-after costs<sup>1</sup>. Because of this, an adequate model for software evaluation

---

<sup>1</sup>In the marketing & acquisition context, morning-after costs denominate unforeseen (hidden) costs that delay the time-to-harvest [33].

must be very flexible to consider many characteristics of software products and allow for the definition of a multitude of intra-portfolio constraints. The list of basic requirements partly corresponds to the criteria set that was used for evaluation of common valuation methods in section 3.3.7.

1. **Structured definition of evaluation criteria**

The model must incorporate a structured and efficient process for determining optimal sets of evaluation criteria that can be used to value the selection candidates' characteristics.

2. **Multiple objectives**

Evaluation of an arbitrary number of objectives must be possible. This should be accomplished without a priori weighting of objectives and without breaking them down to a common scale.

3. **Multiple candidates**

Depending on the domain that software is evaluated of, number of candidates might raise to levels that require inefficiently much computation efforts. Thus, computation of the model should be designed as efficiently as possible to allow efficient evaluation of at least 40 candidates. This limit can be increased later by applying suited heuristic algorithms<sup>2</sup>.

4. **Software portfolios**

Comparison should not be performed between single candidates but between complete portfolios of candidates.

5. **Definition of constraints**

To allow for a detailed modelling of base requirements and candidates, definition of constraints that do not only affect general limitations (e.g. maximum initial costs) but also cover (anti)dependencies between candidates (e.g. software products A cannot be used together with software products B) must be possible.

- a) **Inter-candidate dependencies**

Many software products can either *not* or even *only* be used together with other products. For example, Microsoft Office (usually) cannot be used on Unix operating systems. Thus, possibility of defining dependencies between candidates in a common portfolio is important.

- i. **Dependency**

Candidate A requires candidate B, candidate B might (bidirectional) or might not (unidirectional) require candidate A.

Example (unidirectional): Microsoft Office requires Microsoft Windows but not vice-versa.

- ii. **Exclusion**

Candidate A cannot be member of the portfolio if candidate B is and vice-versa.

Example: Microsoft Windows and Unix should not be selected simultaneously if only one operating system is required.

---

<sup>2</sup>Implementation of heuristic methods lies beyond the scope of this thesis. For examples on how to apply heuristic methods to multiobjective optimisation, see [15] and [28].

b) **Value restriction constraints**

Definition of restrictions for values of specific characteristics (such as *costs*) must be possible.

i. **Upper boundary**

All candidates that, for a certain criterion, have a value higher than the predefined maximum, are excluded from all portfolios.

ii. **Lower boundary**

All candidates that, for a certain criterion, have a value lower than the predefined minimum, are excluded from all portfolios.

c) **Business process support constraints**

As implied by various IT management approaches, IT used in an organisation should as precisely as possible cover all business processes. Because of this, definition of business processes and their coverage by each software candidate is required. Only portfolios in which all relevant business processes are covered by software candidates are feasible choices.

6. **Low complexity**

Obviously, methods with low complexity can be applied easier than those with higher complexity. Because in many companies the step of software valuation is skipped due to a lack of knowledge [58], “simpler” methods are more likely to be employed. Though sometimes complex models are required to represent reality as good as possible, complex models are (in general) more error-prone than less complex ones.

7. **Little “manual” interaction**

The amount of manual interaction required to perform valuation should be minimised. If it is too high, the valuation process can be repeated less frequently, user acceptance will decrease (or even cease) and errors are more likely to happen. If an approach comprises too many manual tasks, the number of candidates and objectives might be limited as manual consideration of too many factors ceases efficiency (e.g. the analytical hierarchical process, section 3.3.7).

8. **Reusability/repeatability**

As the method should not only be used once but regularly, valuation process should be easily repeatable. If there are several similar evaluation processes, collected a priori data should be reusable as far as requirements overlap.

9. **Quality of results**

Of course, quality of results should be as good as possible. Results should be as little as possible blurred and distorted by a priori user preferences. If heuristic methods are applied in later development periods to speed up computation, quality of results must be preserved.

### 4.1.2 Optional Model Requirements

Advanced (“soft”) requirements are supposed to optimise the model regarding its overall performance and usability. Though these requirements will be followed



throughout the phases of model design and implementation, they are not considered crucial for the success (or failing) of the model creation process which is subject to this thesis.

### 1. Usability

The frequency a software is used depends on many factors - one of the most important of them is its usability [60]. Not only but especially because decision makers very often have the free choice of using or not using a software, user experience of the new application should be made as comfortable as possible to leverage acceptance.

#### a) Performance

Time users spend waiting for application start-up and calculation of results should be minimised.

#### b) Comprehensive interface

To reduce time necessary for becoming familiar with the application, its GUI<sup>3</sup> should be as easy to use and as self-explanatory as possible.

#### c) Efficiency

Though the GUI should be kept simple, as few steps as possible should be required to enter a priori data and perform evaluation. A priori data of previous evaluations should be kept (e.g. saved to a file) for later reuse.

### 2. Import of a priori data

Though this approach tempts to minimise necessity of a priori data, capturing a certain amount of information prior to computation process is inevitable. To minimise manual effort, a priori data should be imported from other (already present) data sources such as business process modelling frameworks as much as possible.

### 3. Flexibility

Every domain of business has custom requirements regarding the software employed to support and maintain business processes. This is very important to be kept in mind when developing software that should be applicable for a wide range of business domains. Because of this, no objectives should be “hard-coded”, all parameters must be adaptable to reflect specific pre-conditions and requirements of employing organisations.

### 4. Extensibility

Software, organisations and their respective business itself are subject to continuous change. All developed components should be kept as extensible as possible to permit future adaptation to new concepts and structural changes of the underlying model.

## 4.2 Concept

Valuation and comparison of abstract, intangible or non-measurable characteristics is a problem not only known from software selection. In fact, decision

---

<sup>3</sup>Graphical User Interface

problems in many other - partly related - domains show analogies to and similar types of characteristics (*meta characteristics*). Diversity of characteristics that software can expose is even increased by the fact that the term “software” not only comprises what users mostly associate with it, “classical” software packages (components of-the-shelf). Much more, the term software also applies to many other types of intangible assets that range from operating systems (OS) and database management systems (DBMS) to services (see section 3.2.3) and even custom-tailored software.

In this chapter, a general model will be outlined that combines existing multiobjective portfolio approaches (such as [76], [58], [57]) with the paradigm of business process management finally targeting development of an efficient and good-performance implementation. Formal and logical (mathematical and algorithmic) backgrounds will be described in section 4.3.

### 4.2.1 Definition of Objectives

As mentioned earlier, software criteria subject to optimisation are central component in evaluating software. Therefore, a thorough recognition of requirements and derivation of useful criteria is essential. When defining objectives (see section 3.3.8), care should be taken to cover all relevant characteristics of candidates while keeping the number of objectives within reasonable dimensions<sup>4</sup>. Structured approaches for selection criteria definition show some significant advantages over “intuitive” approaches:

- **Repeatability**

Using data from previous approaches, later analysis rounds can be performed with less effort.

- **Transparency**

Decisions made and preferences expressed in structured approaches are comprehensible and verifiable. It is possible to determine what lead to in- or exclusion of certain criteria.

- **Quality of results**

When deriving criteria from a specified set of base requirements (such as business strategy, application requirements etc.), a maximum of the possible solution space is evaluated.

- **Overall performance**

Structured (scientific) approaches that are either published or have serious intercessors are subject to public discussion. Approaches are criticised, refined or even dropped, and by that continuously optimised. Additionally, structured approaches are likely to deliver results that are suited for use with further specific analysis techniques (such as multiobjective decision making).

---

<sup>4</sup>If the number of criteria is too high, the method might yield too few solutions and computation effort grows inefficient. If it's too low, not all relevant objectives may be covered and the method might yield too many candidates.

The model developed within this thesis uses an adapted OTSO technique to determine an optimal set of software evaluation criteria. Criteria selection is performed by involved managers and other stakeholders, and should be done in form of a workshop. The selection process is performed iteratively in three phases. This means that it is always possible to step back to a previous phase when participants think that it is necessary to refine data collected in this phase.

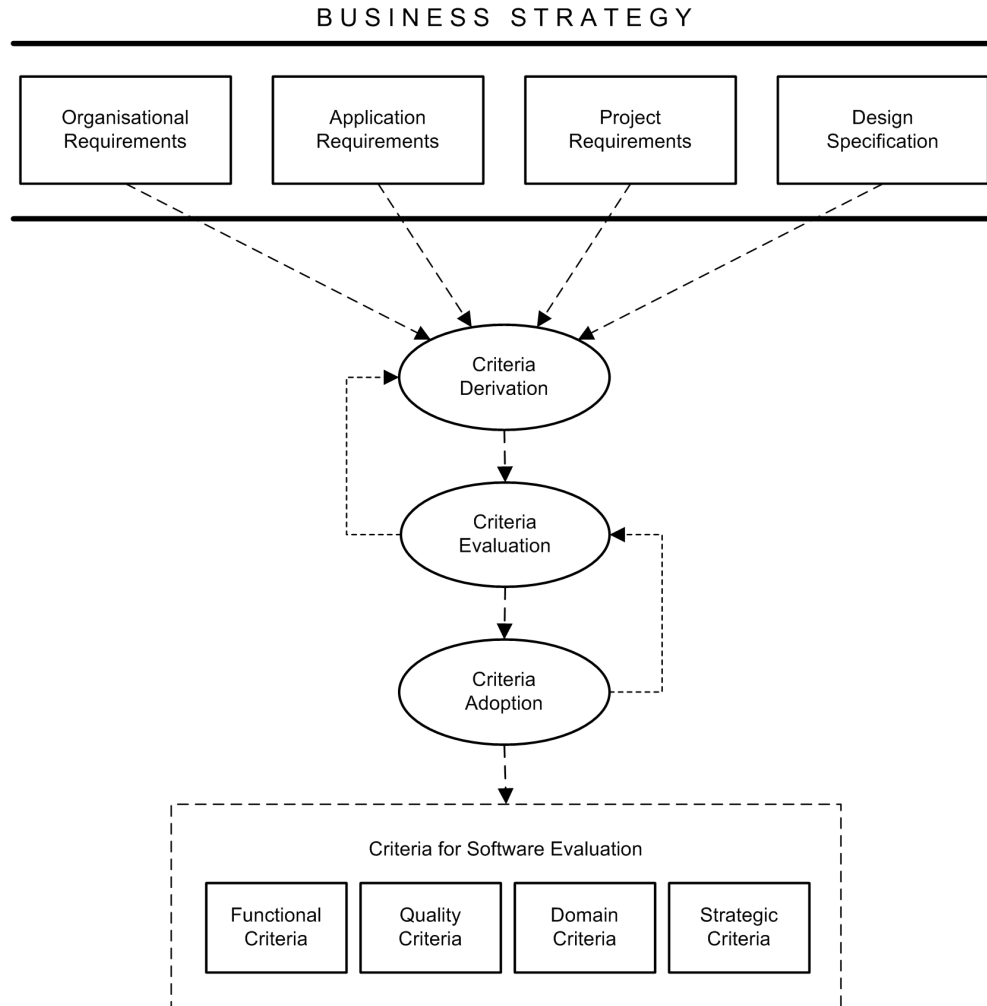


Figure 4.1: Adapted OTSO Criteria Collection Process

In the first phase (*criteria derivation*), requirements are derived from the four main factors. Figure 4.1 visualises the transformation of business strategy to evaluation criteria. Business strategy is represented by the main factors. Note that, though these factors show similarities to those employed in the OTSO model (see section 3.3.8.3), they bear significant differences regarding their meaning and denotation. The four factors are:

- **Application requirements**

In this model, the term of *application requirements* mainly corresponds to the functional requirements a software should be capable of fulfilling. These

requirements are not inductively defined (“what we would like the new software to do”) but deductively collected (“what tasks have to be performed by the software in the company”). By that, the factor of application requirements strongly (but not exclusively) coheres with the company’s business processes. The degree of each software’s business process coverage will be evaluated using criteria derived from this factor.

- **Design specification**

Most organisations already employ IT infrastructure that includes various types of hardware and software. For a flawless and efficient integration into the the company’s IT landscape, new software must be evaluated under the aspect of application design, e.g. interfaces and connectivity, software-architectural constraints, hardware compatibility, availability, extensibility, life-cycle etc.

- **Project requirements**

Here, the term *project plan* does not only refer to the common understanding of business projects<sup>5</sup>, but also to the “project” of introducing new software. Because acquisition of software infrastructure usually requires definition of constraints such as the time frame that the new system should be up and running within, a maximum of initial and subsequent costs, involved stakeholders etc., the term *project plan* refers to the planning of general strategic constraints that affect acquisition, adoption and operation of one or more specific software products.

- **Organisational requirements**

Organisational requirements are the broadest and least tangible factor of all. They refer to many aspects of companies’ general business strategies including long-term targets, external factors, corporate politics etc.

All requirements derived from the main factors are formulated as software evaluation criteria. Requirements that result from a formal basis or specification (such as requirement specifications, system design documents etc.) can be collected directly. This is done either by manual transfer or by automatic import of business processes from BPM tools<sup>6</sup>. Other requirements (especially organisational requirements) can be collected with brainstorming sessions. The larger number of results delivered by brain storming sessions imposes no problem, as number of criteria is reduced in the phase two.

In the second phase (*criteria evaluation*), all collected criteria are evaluated. To keep the number of criteria within reasonable dimensions and to maximise criteria quality, criteria should be reduced by aggregating, combining or dropping them when reasonable and possible. All criteria should be tested regarding their relevancy, quality, granularity and redundancy with other criteria. If this phase unveils that overall criteria quality is too low, or that important requirements and/or criteria have been omitted, participants can step back to the previous phase.

<sup>5</sup>Which is, as undertakings with defined goals, durations and resource consumptions

<sup>6</sup>The software developed in course of this thesis will allow ADONIS models to be imported. See chapter 5.

In practical tests, an overall number of about 5 to 12 criteria has proven reasonable. For each individual decision situation, the optimal number of criteria of course strongly depends on the situation's characteristics and complexity. In any way it is important to realise that, while smaller numbers of criteria still might prove useful for simple decision situations, much larger numbers will significantly slow down computation process.

In the third and final phase (*criteria adoption*), criteria are adopted for use as software evaluation input data. Criteria are matched with the software candidates' characteristics. If there are any characteristics that should be optimised but are not covered by the current criteria set, criteria can be altered by stepping back to phase two. Then, for reasons of simplicity of further analysis, all criteria are classified as *benefit category* or as *resource consuming category*. While benefit categories are value-generating (e.g. cash returns) or at least *positive* (e.g. usability), resource consuming categories are value-consuming (e.g. initial costs) or negative (e.g. required maintenance effort).

All categories are assigned a unit that is suited to express all candidates values in the respective category (e.g. "EUR", "hours" etc.). For abstract and non-measurable categories such as *usability* or *extensibility*, virtual units (like *points*) can be used to express an approximate rating<sup>7</sup>. Therefore, appropriate scales that cover the whole possible parameter bandwidth have to be introduced. For example, if usability should be measured, utilisation of a scale that ranges from 0 points (not usable at all) to 10 points (optimal usability) appears reasonable.

Care must be taken not to confuse "total value categories" (e.g. total initial cost of a portfolio) and "average value categories" (such as usability or performance): Unlike total values such as costs, software portfolio's usability values do not necessarily increase with the number of contained candidates. Thus, in such cases usability must not be represented by the sum of the contained candidates' usability values but by their average value. In addition to this, criteria values can be normalised using quantifiers (such as "1K EUR" for "in units of EUR 1.000") to enable more comfortable and intuitive handling. Figure 4.1 shows an example for a category (criteria) listing.

ID	TYPE	CATEGORY	UNIT
1	Benefit	Usability	Pts.
2	Benefit	Extensibility	Pts.
3	Benefit	Performance	Pts.
4	Resource	Duration of setup	Hours
5	Resource	Initial costs	1K EUR
6	Resource	Maintenance costs	1K EUR
	...	...	...

Table 4.1: Example for resource/benefit categories

<sup>7</sup>Ratings for abstract characteristics can be easily gathered by comparing the relevant characteristic of all evaluation candidates [40]

### 4.2.2 Consideration of Time Periods

The model is capable of considering given input parameters over multiple time periods. This enables volatile objectives to be considered under variation in time. *Initial costs*, for example, usually (but not always, cf. [20]) occur in the first period only (which means that all other periods have a value of zero) while *maintenance costs* usually occur continuously but might be subject to variations (as values of periods differ). Static input parameters that do not change (e.g. usability) constantly keep their value over all periods<sup>8</sup>. See table 4.2.

For time periods, an arbitrary (but constant) duration can be chosen. The length of single periods should be chosen depending on the total time frame that is subject to analysis in a way that there are not more than approximately 8 time periods. More time periods would increase evaluation computation expense and complicate the whole evaluation process without yielding significant gains in result quality. A period duration of one *year* is commonly used as 5 to 8 years of analysis correlate with popular software maintenance and life-cycle intervals (i.e. [54]). If a breakdown in time periods is not necessary, only one time period will be processed and all input parameters will be treated as values of this single period.

### 4.2.3 Collection of Candidate Data

Evaluation of candidates requires collecting all candidates' data regarding all the defined benefit and resource categories (see table 4.2). This data can be acquired either based on facts or on estimations. While data regarding functional criteria usually can be taken from software specifications and manuals, other criteria can only be approximated. Estimation of criteria can be accomplished with open discussions as well as structured decision-making methods such as Delphi processes. Furthermore, approximative candidates data can be obtained using the technique of pairwise comparison and value aggregation as suggested by the AHP (see section 6.2 for an example).

Candidate data that has already been collected can be reused in subsequent evaluations as long as candidates characteristics and categories subject to optimisation do not change.

### 4.2.4 Definition of Restrictions

For each benefit and resource category, restrictions can be defined to limit each portfolio's total value in this category. Restrictions can be defined separately for each time period. This is useful because resource consumption can be shaped to fit availability while benefits can be required to meet minimum expectations. For example, maxima for initial costs and maintenance costs as well as a minimum of performance can be defined (see table 4.3). All portfolios that do not obey restrictions in one or more time periods are dropped only preserving portfolios that match all restrictions.

---

<sup>8</sup>In the implementation, this detail will be hidden to the user.

	CATEGORY	UNIT	P1	P2	P3	P4
CANDIDATE 1						
1	Usability	Pts.	8	8	8	8
2	Extensibility	Pts.	6	6	6	6
3	Performance	Pts.	6	6	6	6
4	Duration of setup	Hours	90	0	0	0
5	Initial costs	1K EUR	36	0	0	0
6	Maintenance costs	1K EUR	5	3	2	2
	...	...	.	.	.	.
CANDIDATE 2						
1	Usability	Pts.	7	7	7	7
2	Extensibility	Pts.	8	8	8	8
3	Performance	Pts.	5	5	5	5
4	Duration of setup	Hours	78	0	0	0
5	Initial costs	1K EUR	31	0	0	0
6	Maintenance costs	1K EUR	0	2	2	2
	...	...	.	.	.	.

Table 4.2: Example for category data over four time periods

When limiting benefit categories, limits are treated as minima. This means that each portfolio's total value must be equal to or higher than a defined limit. When limiting resource categories, limits are treated as maxima. This means that each portfolio's total value must be lower than or equal to a defined limit.

	CATEGORY	UNIT	P1	P2	P3	P4	NOTE
3	Performance	Pts.	7	7	7	7	MIN, Static
4	Initial costs	1K EUR	40	0	0	0	MAX, first phase only
5	Maintenance costs	1K EUR	10	5	5	5	MAX, variable

Table 4.3: Example for category data over four time periods (for two candidates)

#### 4.2.5 Definition of Dependencies

Inter-candidate dependencies are used to establish relationships among software candidates. This is useful because many software products show (inclusion or exclusion) dependencies to others. For example, nearly all applications need an operating system (inclusion) while, in most cases, parallel utilisation of two operating systems or even two office application suites (such as MS Office and Open Office) is not intended (exclusion).

Dependencies are realised using four basic relationships of two different types, *inclusion dependencies* and *value-modifying dependencies* (see below). This concept has been chosen because it permits realisation of nearly all reasonable dependencies by combining two or more basic relationships.

- **Inclusion/exclusion constraints** define whether or not portfolios are valid when containing specific combinations of candidates. This is realised by dropping all portfolios containing combinations of candidates that do not satisfy inclusion/exclusion constraints (see chapter 4.3).
  - **Inclusion**  
At least  $n$  candidates of a list have to be included in each portfolio.
  - **Exclusion**  
Not more than  $n$  candidates of a list can be included in each portfolio
- **Value-modifying dependencies** define positive or negative synergies between candidates in the same portfolio. If a portfolio contains certain pre-defined combinations of candidates, a (positive or negative) sum  $x$  is added to the portfolio's total value of a specified category.
  - **AND (synergy)**  
If at least  $n$  candidates of a list are included, benefits or resources are changed.
  - **OR (anti-synergy)**  
If not more than  $n$  candidates of a list are included, benefits or resources are changed.

Because dependencies can be combined in various ways, and because an arbitrary number of dependencies can be defined, many restrictions can be expressed using the four basic relations. Examples for application of these relations:

- Include at least  $n$  candidates from a list of  $x \geq n$  candidates (inclusion).
- Include not more than  $n$  candidates from a list of  $x \geq n$  candidates (exclusion).
- Include exactly  $n$  candidates from a list of  $x \geq n$  candidates (inclusion combined with exclusion).
- Always include candidate A or candidate B or both (inclusion)
- Include candidate A or candidate B or none of them, but not both (exclusion)
- Always include candidate A or candidate B, but not both (inclusion  $\cap$  exclusion)
- If both candidate A and candidate B are included, value  $d$  is added to the overall value of a specified category (AND).
- If at least  $n$  candidates from a list of  $x \geq n$  candidates are included, value  $d$  is added to the overall value of a specified category (AND).
- If not more than  $n$  candidates from a list of  $x \geq n$  candidates are included, value  $d$  is added to the overall value of a specified category (OR).



- If exactly  $n$  candidates from a list of  $x \geq n$  candidates are included, value  $d$  is added to the overall value of a specified category (AND  $\cap$  OR).

Another important dependency that is applied to portfolios automatically is the constraint of all portfolios' fully covering all business processes (see next section). Those portfolios that do not cover the whole range of defined business processes are dropped. This behaviour has been chosen intentionally as most organisations strive to fully support their business processes with their software<sup>9</sup>.

#### 4.2.6 Definition of Business Process Coverage

As mentioned previously, an important factor in software selection is each candidates' business processes coverage ("Which candidate covers which business processes?"). This data is represented by straight binary n-to-n relations between software products and business processes. This means that software products can either support or not support business processes, but they cannot support a process only "half" or "by 20 percent". If a software product supports only part of a single business process, this process must be split into smaller sub processes that are either fully supported or not.

The design approach of supporting only full process coverages has been chosen intentionally, as blurry information like "software A covers 60 percent of business process X" cannot be seamlessly integrated into a logic model. The point is that, based on the information that "two different software products each support a common business by 60 percent", the extent to which a business process is effectively covered by both software products cannot be determined. If those two software products covered supplementary aspects of the business process (which of course is not possible with a total coverage of 120 percent), the process would be fully covered. If the products both covered the same aspects of the business process, exactly 60 percent of the process would just be "double-covered".

Though this behaviour might, at the first moment, look like a conceptual limitation, the model design permits evaluation of business process coverage with nearly arbitrary granularity: The software candidates' coverages can not only be mapped to whole business processes but also to sub processes and single activities. Various processing units of different granularity can be mixed to represent the system's real requirements. Table 4.4 shows an example of a software-process mapping ("1" means that a software supports a process, "-" means that it does not). Unlike resource and benefit categories, software-process mappings cannot be defined with per-period granularity, a single value per software product and business process is used for the whole evaluation.

Software-process mappings (as well as software-activity mappings) can be represented by a binary  $M \times N$  matrix where  $M$  is the number of software candidates and  $N$  is the total of processes and activities that must be supported.

Because definition of software-process mapping data is highly explicit (due to the binary nature of relationship) and because today many companies use business

<sup>9</sup>Under certain circumstances, it could appear useful to adapt this behaviour to return portfolios that cover at least  $x$  percent of all business processes, or to return all portfolios and include the number of covered processes as decision variable (category). This extension can be implemented easily.

	Process 1	Activity 2.1	Activity 2.2	Process 3	Process 4
Software 1	1	1	-	-	-
Software 2	-	-	1	1	-
Software 3	-	-	1	1	1
Software 4	-	-	1	-	1

Table 4.4: Example for a simple software-process mapping (matrix)

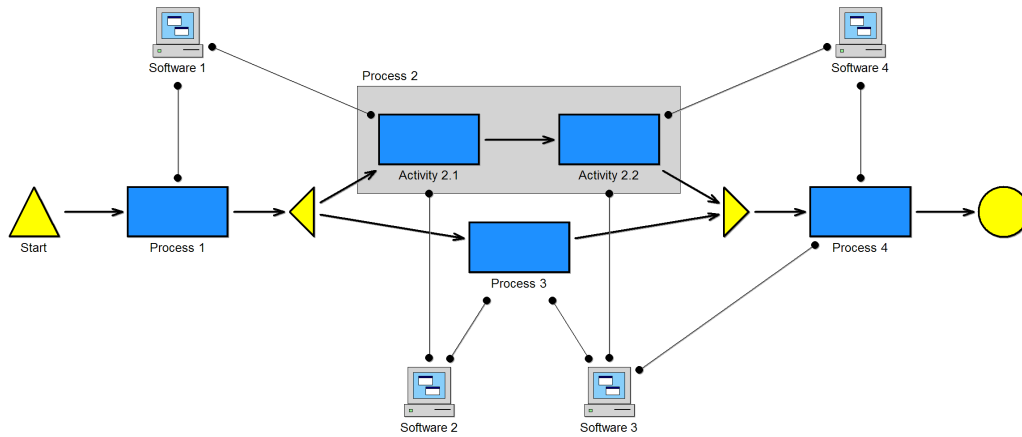


Figure 4.2: BPM Containing Software-Process Mappings

process modelling (BPM) software to model their business processes, it appears efficient to directly import data regarding relations of software and processes from BPM models. Figure 4.2 shows a BPM model representation of the mapping data in table 4.4.

As this thesis emphasises the benefits of holistic IT management approaches, the integration of business process management into decision support for software selection is demonstrated by implementation of a specific BPM import routine. This routine permits import of BPM models created with the commercial business process modelling tool ADONIS<sup>10</sup>. For further details on the ADONIS data import, see chapter 5.4.

## 4.3 Formalising the Model

In this section, the model will be formalised to permit a straight-forward implementation. Though the formal approach shows analogies to other multiobjective portfolio approaches (such as the S/H approach), most model components have been adapted and domain-specific concepts (such as business process coverage) have been added to satisfy the requirements of software selection.

### 4.3.1 Creation of Portfolios

Creation of portfolios is performed automatically based on given input data. Solution space is filled using complete enumeration: All  $C^C - 1$  possible portfolios

<sup>10</sup>Version 3.81 of the *ADONIS Geschäftsprozessmanagement Toolkit* from BOC GmbH

$P$  containing between 1 and  $C$  software candidates  $c \in C$  are enumerated. For each current portfolio  $p \in P$ , the indicator  $\alpha_{p,c}$  denotes whether the candidate is included ( $\alpha_{p,c} = 1$ ) or not ( $\alpha_{p,c} = 0$ ). The number of candidates contained in a certain portfolio can be determined by

$$|C_p| = \sum_{i=1}^C \alpha_{p,c_i} \quad (i = 1, \dots, |C|) \quad (4.1)$$

### 4.3.2 Time Periods

Because all portfolios must be evaluated in multiple time periods, all calculations must be broken down and performed for multiple time intervals. A parameter  $t$  is introduced that specifies the current of all  $T$  time periods. Every time period is evaluated and treated separately.

### 4.3.3 Benefit/Resource Categories

Categories  $a \in A$  are divided into benefit categories  $B \subset A$  and resource categories  $R \subset A$  (with  $|B| + |R| = |A|$ ). All candidates have specific values  $v_{c,t,a}$  for all categories and time periods  $t \in T$ . For every portfolio in a certain time period, the total value over all included candidates for a specific category  $a$  is called the *portfolio's category value*  $u_{p,t,a} \in U$  for that category. Category values are main objectives for all further evaluations of portfolios.

For a given category and time period, a portfolio's *total* category value  $u_{p,t,a}$  is calculated as follows:

$$u_{p,t,a} = \sum_{i=1}^{|C|} (\alpha_{p,c_i} v_{c_i,t,a}) \quad (i = 1, \dots, |C|; t = 1, \dots, T) \quad (4.2)$$

In analogy to this, a portfolio's *average* category value  $u_{p,t,a}$  is calculated by:

$$u_{p,t,a} = \frac{\sum_{i=1}^{|C|} (\alpha_{p,c_i} v_{c_i,t,a})}{|C_p|} \quad (i = 1, \dots, |C|; t = 1, \dots, T) \quad (4.3)$$

### 4.3.4 Dependencies

There are two different types of relationships among candidates:

#### 4.3.4.1 Inclusion/Exclusion Constraints

Inter-candidate constraints  $k \in K$  are divided into inclusion constraints  $L \subset K$  and exclusion constraints  $E \subset K$  (with  $|L| + |E| = |K|$ ). Each of these constraints consist of a list of  $n \leq |C|$  candidates as well as a value that defines the minimum (for inclusions) or maximum (for exclusions) of candidates on this list that must be contained in a common portfolio to satisfy the constraint. Function  $f_{(p,k)}$  determines whether a portfolio  $p$  satisfies a constraint ( $f_{(p,k)} = 1$ ) or not ( $f_{(p,k)} = 0$ ).

Portfolios are dropped if they either a) do not satisfy *all* inclusion constraints or b) satisfy *one or more* exclusion constraints. This is determined for inclusion and exclusion constraints separately:

Inclusion constraints:

$$f_{(p,k)} = 1 \quad \forall k \in L \quad (4.4)$$

or

$$\sum_{i=1}^{|L|} f_{(p,k_i)} = |L| \quad (i = 1, \dots, |L|; k_i \in L) \quad (4.5)$$

Exclusion constraints:

$$f_{(p,k)} = 0 \quad \forall k \in E \quad (4.6)$$

or

$$\sum_{i=1}^{|E|} f_{(p,k_i)} = 0 \quad (i = 1, \dots, |E|; k_i \in E) \quad (4.7)$$

As composition of portfolios does not change over time periods, these inclusion/exclusion constraints are time-independent.

#### 4.3.4.2 Value-Modifying Dependencies

Value-modifying dependencies  $o \in O$  are divided into AND and OR dependencies. They are defined in analogy to inclusion/exclusion constraints: Each dependency consists of a list of  $n \leq |C|$  candidates as well as a value that defines the minimum (for AND dependencies) or maximum (for OR dependencies) of candidates on this list that must be in a common portfolio for the dependency to exist in this portfolio. If a dependency exists within a portfolio, this dependency's (positive or negative) category value  $q_{t,o}$  is added to the respective category value of this portfolio for the current time period.

Function  $g_{(p,o)}$  determines whether a dependency exists within a portfolio ( $g_{(p,o)} = 1$ ) or not ( $g_{(p,o)} = 0$ ). The sum of dependency values  $q$  that are added to a portfolio's category value  $a$  in time period  $t$  can be calculated as

$$\Delta u_{p,t,a(O)} = \sum_{i=1}^{|O|} g_{(p,o_i)} q_{t,o} \quad (i = 1, \dots, |O|) \quad (4.8)$$

By that, a portfolio's category value for a certain time period can be interpreted as

$$u_{p,t,a} = \sum_{i=1}^{|C|} (\alpha_{p,c_i} v_{c_i,t,a}) + \sum_{i=1}^{|O|} g_{(p,o_i)} q_{t,o} \quad (t = 1, \dots, T) \quad (4.9)$$

### 4.3.5 Benefit/Resource Restrictions

For all categories in all time periods, specific restrictions  $x_{t,a} \in X$  can be defined that limit the minimum (benefit category) or maximum (resource category) of each of the portfolios' category values in each time period. All Portfolios that do not satisfy the following inequations are dropped.

Benefit restrictions:

$$u_{p,t,a} \geq x_{t,a} \quad \forall t \in T \quad (a \in A) \quad (4.10)$$

Resource restrictions:

$$u_{p,t,a} \leq x_{t,a} \quad \forall t \in T \quad (a \in A) \quad (4.11)$$

### 4.3.6 Business Process Coverage

The coverage of business processes by portfolios is realised in analogy to inclusion constraints. For every business process  $s \in S$ , the function  $w_{(c,s)}$  determines whether a process is covered by a candidate ( $w_{(c,s)} = 1$ ) or not ( $w_{(c,s)} = 0$ ). By that, a portfolio's support for a specific process (i.e. the portfolio's containing at least one candidate which supports the process) can be determined by

$$w_{(p,s)} = [OR]_{i=1}^{|C(p)|}(w_{(c_i,s)}) \quad (c_i \in p; w_{(p,s)} \in \{0, 1\}) \quad (4.12)$$

The number of processes covered by a portfolio can be determined by

$$\vec{w}_{(p,S)} = s \in S \times [OR]_{i=1}^{|C(p)|}(w_{(c_i,s)}) \quad (c_i \in p; w_{(p,S)} \in \{\{0, 1\}, \{0, 1\}, \dots\}) \quad (4.13)$$

Because support of *all* business processes is a knock-out constraint in this model, all portfolios are dropped that do not satisfy the following equation:

$$\sum_{i=1}^{|S|} [OR]_{j=1}^{|C(p)|}(w_{(c_j,s_i)}) = |S| \quad (c_i \in p; s_i \in S) \quad (4.14)$$

## Chapter 5

# Implementation

To test the software selection model created in the previous chapter, a decision support system will be implemented that automates the search for efficient portfolios. Because collection of decision situation data is a major step in solving decision problems, user interface must be transparent and easy to use. Because the application should be capable of solving many different (and partly quite complex) types of software selection scenarios, calculation algorithms must be designed with focus on efficiency, performance and - of course - coverage of all the previously stated functional model requirements. To improve readability, the application will from now on be referred to as “MultiSel”.

To permit system independent and simultaneous access to MultiSel, the concept of a web application was chosen. All data is entered under use of web forms, subsequent analysis is performed on the server itself. This design does not only permit clients with little computation power to use the software efficiently but also enables usage from remote locations. Thus, provided that the application server itself bares sufficient computation power, even operation of a central evaluation internet service appears possible.

Among the many possible development platforms enabling construction of interactive web applications, Microsoft .NET Framework 2.0 has been chosen to implement MultiSel. It permits a straight-forward implementation and allows writing consistent, cohesive code that can be subsequently extended and adapted to many other application scenarios.

ADONIS models have been chosen as source for component and business process imports because ADONIS natively permits graphical modelling of workflows (consisting of single business processes/activities) and - which is of most importance - mapping activities to available resources (like software components). In addition to this, the used version of ADONIS allows export of models (diagrams) as well-formed and valid XML files that can be easily parsed and imported into the MultiSel database.

## 5.1 Data Model

Data model of MultiSel (shown in figure 5.1) has been kept as generic as possible to allow for a later adaptation to changing requirements and reuse for selection problems of other domains. Data tables used to store a-priori solution data which

is supposed to be analysed are printed in blue, tables containing the output of the decision situations' evaluation runs (a-posteriori data) are printed in green. Table 5.1 contains short descriptions of all data objects.

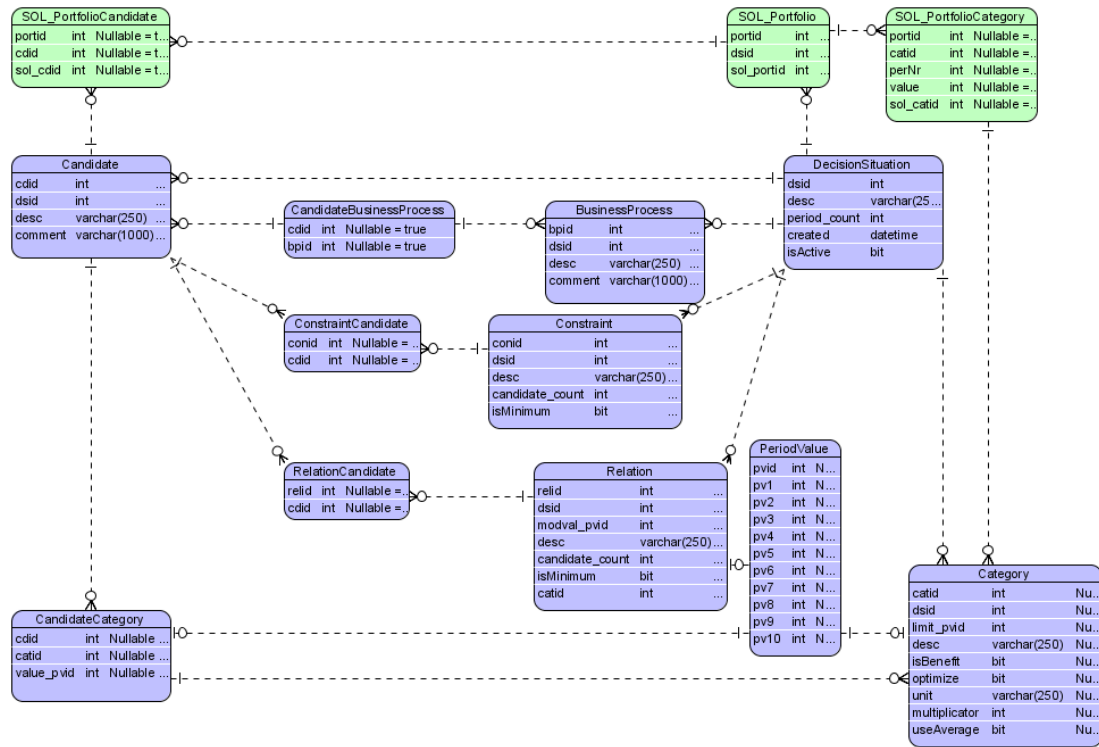


Figure 5.1: The complete MultiSel data model

## 5.2 Application Design

Beyond functionality, usability is an important quality factor for most types of software. To leverage user acceptance and permit efficient and transparent collection of input data, particular attention is paid to the graphical user interface (GUI).

### 5.2.1 User Interface

The MultiSel application consists of three major interfaces which are employed sequentially to perform an evaluation:

- A data input interface serves for entering solution data based on which an evaluation can be performed.
- An evaluation component is used to perform evaluation of decision situation input data and determine efficient portfolios.
- A graphical solution explorer application that permits finding optimal portfolios by dynamical application of category limit restrictions.

Object	Description
BusinessProcess	Business processes that candidates are mapped to.
Candidate	Candidates that are to be evaluated within a decision situation.
CandidateBusinessProcess	N to N relation between candidates and their covered business processes.
CandidateCategory	N to N relation between candidates, categories and period values. This table specifies the period values for each candidate in each of the decision situation's categories.
Category	All categories subject to analysis.
Constraint	Inclusion/exclusion constraints among candidates.
ConstraintCandidate	N to N relation between constraints and candidates.
DecisionSituation	All decision situations.
PeriodValue	Category values for all periods.
Relation	AND/OR dependencies among candidates.
RelationCandidate	N to N mapping between dependencies and candidates.
SOL_Portfolio	Efficient portfolios that have been determined.
SOL_PortfolioCandidate	N to N relation between portfolios and all contained candidates.
SOL_PortfolioCategory	N to N relation between portfolios and their respective candidates.

Table 5.1: Objects in the MultiSel Data Model



### 5.2.1.1 Data Input Interface

Solution data is entered in five steps. In the **Sessions mask**, a new decision situation can be created. A name for the decision situation as well as the required number of periods can be specified (see figure C.1).

In the **Categories mask**, the objectives that candidates are evaluated by can be defined. Together with a description, the categories' types (benefit or resource) as well as their unit and multiplier can be defined here. Additionally, for each category a flag can be set that defines whether or not the category should be optimised (see appendix C, figure C.2).

In the **Components mask**, candidates that are primary subject to evaluation can be defined. For all components, a description and their period values for all categories can be entered. This data is very central because it determines how much benefits a component yields and how much resources its usage requires. Components can be either entered manually or imported from an ADONIS business process model (see appendix C, figure C.3).

In the **Constraints/Relations mask**, inclusion/exclusion constraints and AND/OR dependencies are defined. Upper/lower limits for all resource/benefit categories can be defined for each single period (see appendix C, figure C.4).

In the **Business Processes mask**, business processes and the mapping of all candidates to one or more business processes can be defined. Business processes and the mapping of candidates to business processes can be either defined manually or imported from an ADONIS business process model (see appendix C, figure C.5).

### 5.2.1.2 Evaluation Component

The MultiSel evaluation component, which is built in analogy to [59], is engaged using the **Evaluation mask** (see appendix C, figure C.6). It performs the actual data analysis and writes back all solution data into the MultiSel database.

The connection to the INNOV library is realised using ASCII text files. When an evaluation is started, decision situation data from database is aggregated and written to an ASCII text file in a proprietary format<sup>1</sup>. After this, the INNOV library is invoked to process the given input file. It generates an output file containing all found solutions (efficient portfolios of candidates) which is read back into the database.

## 5.2.2 Limitations of Implementation

As the current implementation of the MODS software selection model mainly serves as a proof of concept, certain limitations of functionality have been applied. Due to restrictions of the INNOV library and the interface between this library and the MultiSel data environment, MultiSel supports a maximum of 5 periods and a maximum of 32 candidates. For information on additional functionality that could be integrated into both the model and the MultiSel application, refer to section 8 (Directions for Further Research).

---

<sup>1</sup>See the INNOV library documentation for more information [3].

## 5.3 Stummer's INNOV Library

The INNOV library was developed in course of Stummer's writings of his dissertation in 1997. Stummer successfully created a multiobjective Pareto model that permits selection of non-dominated R&D projects. The INNOV library was created to solve the problem of enumerating and finding all non-dominated portfolios. Because the problem of finding efficient project investment strategies strongly adheres to the problem of finding efficient software portfolios, this library is well-suited for an adaptation to the model of software selection constructed in this thesis.

Because the library was written in C++, its execution performance is considerably better than that of higher-level languages such as Java or .NET: Up to  $2^{30}$  different portfolios can be analysed within feasible time on a standard workstation, meaning that a total of 30 candidates can be analysed using today's standard hardware. The INNOV library uses ASCII files for data input and output. It is called as a Shell application with command line parameters and does not include a graphical interface. The INNOV library's proprietary input data format had to be extended in order to allow for consideration of business processes and modelling of the mapping of candidates and business processes. For more information on the INNOV library, see the INNOV library documentation [3].

### 5.3.1 Extension of Input Data Format

The INNOV input data format had to be extended due to the reasons mentioned previously. Listing 5.1 shows a snippet of sample input data<sup>2</sup> that has been enriched with additional symbols to store business processes and candidate to business process mappings. A new data row type was introduced (see table 5.1, lines 5 and 53) that denotes which business processes are covered by the current candidate. The line consists of separated binary digits  $d$  that have a value of  $d(i) = 1$  if the current candidate covers the business process on position  $i$  (from a total of  $I$  business processes) or  $d(i) = 0$  if the candidate does not.

```

32 1 Component_1 # Name
33 4 # Laufzeit von Prj 1
34 0 1 0 0 1 0 # NEW: COMPONENT MAPPING OF BUSINESS PROCESSES
35 100 200 300 400 0 # Nutzenkat. 1
36 0 0 0 0 0 # minimum N1
37 0 0 0 0 0 # maximum N1
38 1 2 3 4 0 # Nutzenkat. 2
39 0 0 0 0 0 # minimum N2
40 0 0 0 0 0 # maximum N2
41 150 300 400 500 0 # Ressourcenk. 1
42 0 0 0 0 0 # minimum N1
43 0 0 0 0 0 # maximum N1
44 1 1 1 1 0 # Ressourcenk. 2
45 0 0 0 0 0 # minimum N2
46 0 0 0 0 0 # maximum N2
47 2 2 2 2 0 # Ressourcenk. 3
48 0 0 0 0 0 # minimum N3
49 0 0 0 0 0 # maximum N3

```

<sup>2</sup>Taken from the INNOV documentation [3].

```

50 2 Component_2 # Name
51 5 # Laufzeit von Prj 2
52 1 0 0 1 1 0 # NEW: COMPONENT MAPPING OF BUSINESS PROCESSES
53 100 200 300 400 500 # Nutzenkat. 1
54 0 0 0 0 0 # minimum N1
55 0 0 0 0 0 # maximum N1
56 1 2 3 4 5 # Nutzenkat. 2
57 0 0 0 0 0 # minimum N2
58 0 0 0 0 0 # maximum N2
59 150 300 400 500 400 # Ressourcenk. 1
60 0 0 0 0 0 # minimum N1
61 0 0 0 0 0 # maximum N1
62 9 8 7 6 5 # Ressourcenk. 2
63 0 0 0 0 0 # minimum N2
64 0 0 0 0 0 # maximum N2
65 2 1 2 1 2 # Ressourcenk. 3
66 0 0 0 0 0 # minimum N3
67 0 0 0 0 0 # maximum N3

```

Listing 5.1: Extended INNOV Input Data: Business Processes

In addition to this, the input file header has been adapted to contain the total number of business processes (see listing 5.2, line 2).

```

1 MULTISEL – Example File
2 4 3 4 5 6 # Number of: candid. / ben. cat. / res. cat. /
3           #           periods / BPs.

```

Listing 5.2: Extended INNOV Input Data: Header

For more information regarding the INNOV library input data format, see the INNOV library documentation [3].

### 5.3.2 Extension of Output Data Format

INNOV output data has been extended to contain the number of business processes involved. An example for extended output data is shown in table 5.3.

```

1 MULTISEL ASCII-OUT
2 Nb_cdi : 4 # number of candidates
3 Nb_res : 4 # number of res. cat.
4 Nb_prof: 3 # number of ben. cat.
5 Nb_per : 5 # number of periods
6 Nb_bp  : 6 # NEW: number of BPs
7 Nb_sols: 6 # number of solutions (portfolios)
8 14 # number of efficient portfolio
9 Prof0 310 620 930 850 630 # portfolio data...
10 Prof1 14 28 42 55 26
11 Prof2 13 25 35 47 7
12 Res0 465 930 1240 1350 1260
13 Res1 47 47 55 52 54
14 Res2 17 14 21 22 26
15 Res3 28 29 30 35 40
16 13 # number of efficient portfolio
17 Prof0 310 620 930 850 130 # portfolio data...
18 Prof1 3 6 9 12 10
19 Prof2 13 24 35 46 6
20 Res0 465 930 1240 1350 860

```

```

21 Res1 24 25 35 36 36
22 Res2 17 16 18 19 19
23 Res3 30 30 30 30 20

```

Listing 5.3: Extended INNOV Output Data (Partial)

## 5.4 Import of ADONIS Models

Business process modelling tools such as ADONIS enable companies to model (and by that standardise) their workflows, business processes and activities. Because collection of information regarding type and characteristic of a company's business processes very often implies the use of BPM tools, an import of this model data into MultiSel appears both reasonable and efficient.

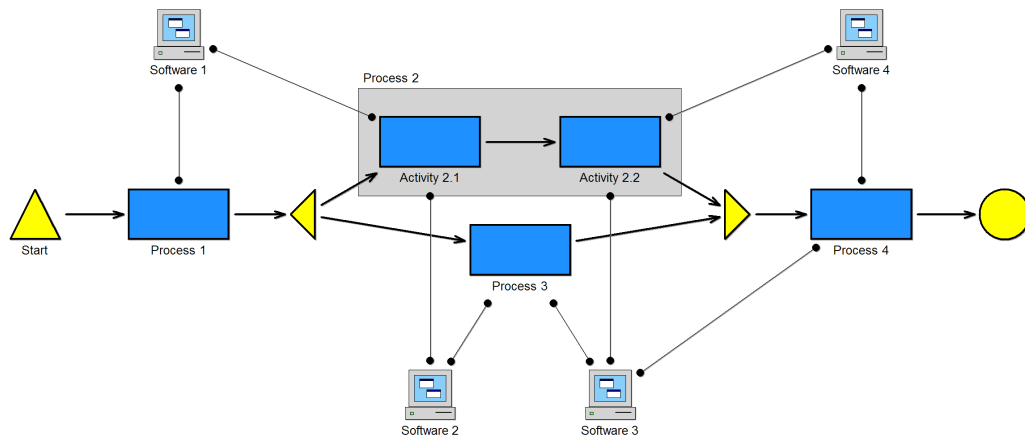


Figure 5.2: ANDONIS Business Process Model

If such models do not only contain information about processes but also about related resources such as required software, this information can be used not only to define processes but also to specify relevant business processes and mappings.

Figure 5.2 shows the business process model presented in the previous section. Using the ADONIS *XML Export* function, this model can be exported as XML file (see listing 5.4). ADONIS XML files can be imported easily because data consists of only three different entity classes: Activities as well as resources are represented by *INSTANCE* elements, references are represented by *CONNECTOR* elements.

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE ADOXML SYSTEM "adoxml31.dtd">
3 <ADOXML version="3.1" adoversion="Version_3.81">
4   <MODELS>
5     <MODEL>
6       <MODELATTRIBUTES>
7         <!-- Attributes -->
8       </MODELATTRIBUTES>
9       <INSTANCE id="obj.11" class="Aktivität" name="BProcess_1">
10        <!-- Attributes -->
11      </INSTANCE>
12      <INSTANCE id="obj.13" class="Ressource" name="Component_A">
13        <!-- Attributes -->

```

```

14     </INSTANCE>
15     <CONNECTOR id="con.21" class="verwendet">
16         <FROM instance="BProcess_1" class="Aktivität"></FROM>
17         <TO instance="Component_A" class="Ressource"></TO>
18         <!-- Attributes -->
19     </CONNECTOR>
20 </MODEL>
21 </MODELS>
22 </ADOXML>

```

Listing 5.4: ADONIS XML File (Partial)

By that, *candidates, business processes (activities)* as well as the *mapping between them* can be imported into the MultiSel model with only a few clicks. During import, objects (candidates and business processes) that already exist in the model (using their description as differentiation criteria) are not overwritten. This allows to update the MultiSel model with newer BPM versions without loss of any changes that have been made to the model (as long as objects just keep their original description).

ADONIS does not allow multiple objects with the same name, every object's name has to be unique. As extensive models for reasons of understandability and clarity often require the use of more than one instance of certain objects (which, in our specific case, are software components), a method had to be found that allows the use of multiple instances of the same software in business process models and a subsequent recognition and association of equivalent objects during the MultiSel import.

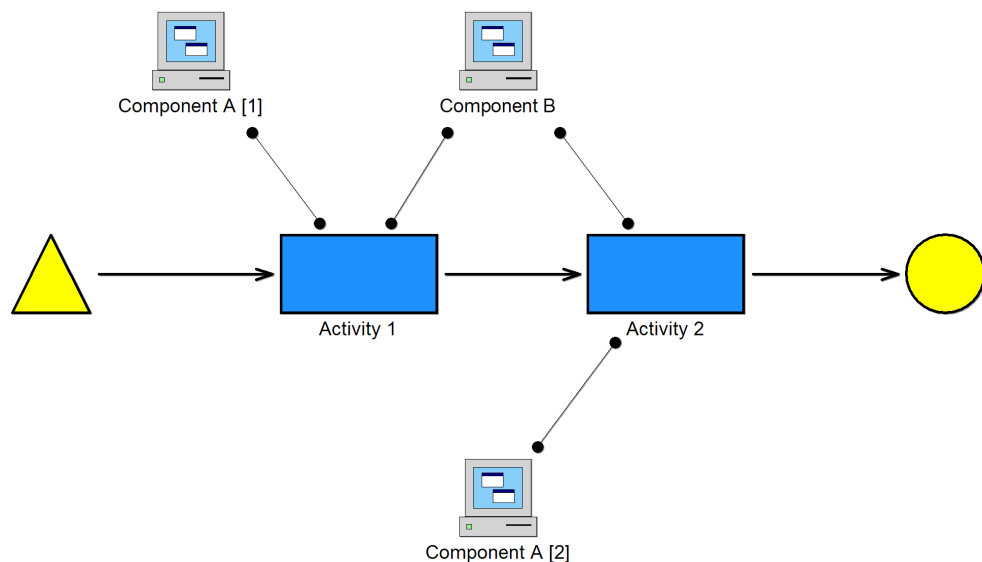


Figure 5.3: Naming of Multiple Software Instances in Common BPMs

This is accomplished by permitting a standardised naming suffix to be used in ADONIS models which is recognised and removed at import time. Figure 5.3 shows a simple business process model which includes two instances of the software *Component A*. Both instances are named using the object's original name ("Component A") followed by a unique integer index enclosed in square brackets

(“[1]”, “[2]”). As soon as the model is imported into MultiSel, all instances of *Component A* are merged to a single component named *Component A* and all relations of the two instances to activities are remapped to this component.

The chosen model of searching for portfolios that cover *all* business processes does not permit business processes to not depend on any components (they would be considered “uncovered” for all portfolios, so no valid portfolios could be found at all). Thus, business processes that do not require any software support must be associated to a dummy component. For this dummy component, a new component can be added that does not have any category values (always zero) and is not included in any constraints or dependencies. Note that this component must have the word “dummy” in its description as it has to be recognised by the software to be excluded from average calculations. When calculating category averages, a category value of zero would diminish the calculated average.

## Chapter 6

# Comparative Case Study

Functionality and result quality of new methods must be measured in order to obtain an idea of how *good* they really are. For algorithms that predict real-world results under use of a model, testing can be done by using real or semi-real<sup>1</sup> input data and comparing results to those of other, related algorithms. In this chapter, capabilities of MultiSel will be compared to those of two other software selection approaches.

Because the Analytical Hierarchic Process is frequently used in various selection scenarios, and because it has been applied to the domain of software selection very often in the past, it was chosen as the first approach to be compared to the new MODS model. Because even today, many organisations still perform software selection using aged cost-benefit calculation techniques, a weighted scoring method as described in section 3.3.3.1 was chosen as the second approach the model will be compared with.

## 6.1 Input Decision Situation

As mentioned in previous chapters, quality of results yielded by decision-supporting algorithms substantially depends on quality of applied input parameters. As these parameters can be subject to notable variance, decision situations that aim at testing and/or comparing algorithms should not only provide balanced overall input parameters but also cover as many special cases as possible [12]. As fictive input data can be tuned to cover big ranges of possible variations, performance of all methods can be tested under optimal evaluation conditions. Thus, decision situation that algorithms are applied to will be derived from a real decision situation, but input parameters will be adapted to reflect a wider range of characteristics.

### 6.1.1 Base Characteristics

A big health insurance company is looking for a way to leverage efficiency of processing incoming requests for reimbursement of medical costs. Therefore, a business process model is generated that defines the whole internal workflow from

---

<sup>1</sup>Input parameters are taken from a real-world decision situation but adapted manually to emphasise specific characteristics.

reception of a request over request approval until informing respective applicants (see figure 6.1). A list of all activities is depicted in table 6.1.

Nearly all activities involved in request processing require permanent support of underlying IT infrastructure. Thus, software selection has to be performed with a main focus on functional activity coverage.

NR	DESCRIPTION
1	Register Request
2	Check request for eligibility
3	Return request to applicant
4	Scan request
5	Check for identical request
6	Open request
7	Open document
8	Edit document
9	Mark for approval
10	Open marked request documents
11	Edit request
12	Approve request
13	Check correctness
14	Inform applicant

Table 6.1: Activities Performed in Request Processing

### 6.1.2 Evaluation Criteria

The criteria set that serves as main measurement objective will be collected using the adapted OTSO method introduced in section 4.2.1). This implies that the company's global and specific business strategies have to be expressed as constraints and parameters of the model's four main selection factors (organisational requirements, application requirements, project plan and design specification).

In the following section, constraints and preconditions of each sector will be analysed and used to derive adequate selection criteria. Note that many general criteria that lead to a sudden exclusion of a candidate (such as "Software must run under Microsoft Windows") are not mentioned explicitly as this thesis focuses on the challenge of combining and comparing non-binary characteristics to find optimal solutions. For each category, attributes "Scale-Min." and "Scale-Max." define the minimum and maximum value of the applied scale *per candidate* (not per portfolio).

#### 6.1.2.1 Application Requirements

A big part of requirements for a software landscape are derived from application requirements. These requirements not only include coverage of relevant business processes but also general characteristics such as specific behavioural, functional and non-functional constraints.



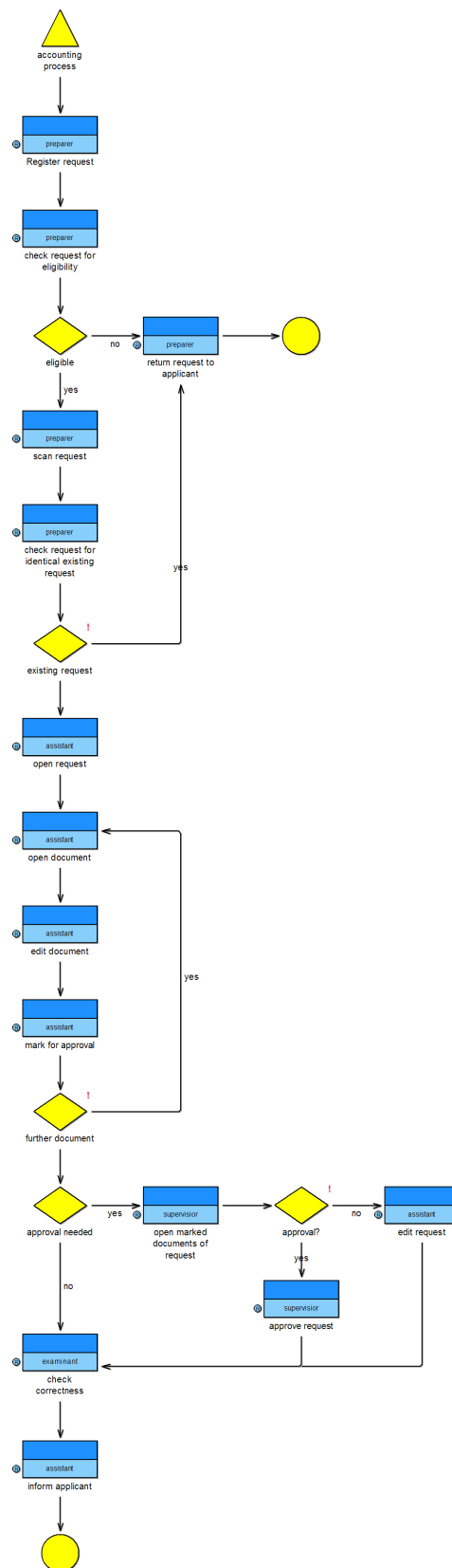


Figure 6.1: BPM: Processing of Incoming Requests

While most requirements regarding functionality usually can be expressed by mapping of activities (or business processes) to candidates, some (such as usability, performance or fault tolerance) cannot. By that, beyond the mapping of business processes and candidates, an optimisation of candidates' usability and fault tolerance is required. Both characteristics will be measured on a scale analogue to that previously defined for the criterion *access control*.

DESCRIPTION	UNIT	SCALE-MIN	SCALE-MAX
BP Coverage	No. of BPs	0	-
Usability	Pts.	0	10
Fault Tolerance	Pts.	0	10

Table 6.2: Criteria Set Derived from Application Requirements

### 6.1.2.2 Design Specification

Because all server-side applications are virtualised, hardware consumption of services can be measured by counting the average number of CPUs a service requires to function properly. This information can be obtained either through manufacturers' specifications or by performing preliminary system test.

Reusability is an important criterion for establishing an efficient software landscape. While, for example, web services usually are highly reusable, reusability of proprietary client-side applications significantly varies and thus must be determined on a per-case basis.

While hardware consumption can be measured in a real value (CPUs), a measurement scale for reusability must be defined manually.

DESCRIPTION	UNIT	SCALE-MIN	SCALE-MAX
H/W Consumption	CPUs	0	-
Reusability	Pts.	0	10

Table 6.3: Criteria Set Derived from Design Requirements

### 6.1.2.3 Project Requirements

The basic conditions that guide acquisition of new IT infrastructure can be treated as a project. For the software selection process that leads to establishment of a new, more efficient request processing workflow, maxima for multiple factors are defined. Expenses such as initial costs, duration of implementation, human resources involved or subsequent (maintenance) costs are restricted and their occurrence should be minimised.

System maintenance is performed in bi-monthly intervals that the new software's maintenance intervals should correspond to as far as possible. Thereby, maintenance intervals of multiples of two months are feasible as well (because in this case maintenance can be performed just e.g. every second time). Thus, a scale is defined that defines the degree of the candidates' maintenance interval

lengths' deviations from a bi-monthly interval, where 0 means "no deviation" and 30 means "30 days deviation". This criterion, of course, has to be minimised.

In addition to this, criteria of initial costs (in 1.000 EUR), maintenance costs (in 1.000 EUR p.a.) and setup duration (in working days) are defined.

DESCRIPTION	UNIT	SCALE-MIN	SCALE-MAX
Initial Costs	1.000 EUR	0	-
Maintenance Costs	1.000 EUR	0	-
Setup Duration	Working Days	0	-

Table 6.4: Criteria Set Derived from Project Plan

#### 6.1.2.4 Organisational Requirements

As a big national carrier of health insurance contracts, corporate policy requires all employed software to enforce specific minimum levels of audit trail and access control. Capability of software candidates regarding these requirements varies as certain software candidates have (partly extensive) built-in logging and access restriction mechanisms while others must be extended before meeting minimum requirements. To allow for the measurement of each of the candidate's specific functionalities in this context, specific scales with a measurement unit of *points* are defined for the combined requirement of *access control*. The scale for this requirement ranges from 0 points (no audit trail, no ACLs) to 10 (all transactions are logged and can be rolled back, repeated and traced to all involved human and infrastructural sources, all accesses are controlled via ACL).

Because processing requests and applications are integral components of core business, and because delays due to hardware or software malfunctions are considered unacceptable, reliability (stability) is a main criterion that will be considered in analogy to requirements of audit trail and access control.

DESCRIPTION	UNIT	SCALE-MIN	SCALE-MAX
Access Control	Pts.	0	10
Reliability	Pts.	0	10

Table 6.5: Criteria Set Derived from Organisational Requirements

#### 6.1.2.5 Final Criteria Set

The aggregation of the four main factors' criteria set leads to definition of an overall criteria set (see table 6.6). For some categories, hard limits for portfolios' maximum total value have been defined. Depending on whether the portfolios' category values should be maximised or minimised, the category type *benefit* or *resource* is assigned. Moreover, all categories must be declared to be analysed per portfolio regarding *total value* or *average value* of all contained components. If requirements are redefined during further evaluation, criteria set can be adapted at any time to optimally fit the company's needs.

SHORT	DESCRIPTION	UNIT	LIMIT	TYPE	ANALYSIS
AC	Access Control	Pts.	-	Benefit	Average
BC	BP Coverage	No. of BPs	-	Benefit	Total
FT	Fault Tolerance	Pts.	-	Benefit	Average
HW	H/W Consumption	CPUs	-	Resource	Total
IC	Initial Costs	1.000 EUR	$\leq 140$	Resource	Total
MC	Maintenance Costs	1.000 EUR	$\leq 20$	Resource	Total
RL	Reliability	Pts.	-	Benefit	Average
RU	Reusability	Pts.	-	Benefit	Average
SD	Setup Duration	Working Days	$\leq 135$	Resource	Total
US	Usability	Pts.	-	Benefit	Average

Table 6.6: Final Set of Selection Criteria

### 6.1.3 Software Candidates

Prior to evaluating candidates using structured approaches, an initial set of feasible evaluation candidates must be defined. This is usually done by performing a rough selection of potential candidates and comparing their main characteristics to the decision situation's base line parameters (i.e. knock-out criteria) such as required operating system, approximate available monetary or time-related resources, application domain etc.

The business process model shown in figure 6.1 requires the following business software components for all its processes to be covered:

- Document Management System (DMS)
- Paper Digitising/Scanning Application
- Data Access Component
- Accounting
- Archiving

The number of candidates to include in individual evaluation strongly depends on several factors including application domain and availability of suited software. Including too many candidates can render evaluation process very complex and significantly increase calculation time of automatic decision support algorithms (such as MultiSel). Including too few candidates can lead to a small solution space resulting in too few, presumably sub-optimal solutions. According to these preconditions and the requirements of the business process model, feasible components have been chosen (see table 6.7).

Depending on the number of applied categories, time periods and business processes, MultiSel is capable of evaluating up to 30 candidates per decision situation. Thus, a balance between all determining variables has to be found. The current decision situation (which includes 9 categories plus business coverage, 3 periods, 14 business processes and 18 software candidates) can be solved on an average workstation in less than 10 seconds.

NAME	FUNCTIONALITY	DESCRIPTION
Component A	Paper Digitising	Existing scan software
Component B	Data Access Component	WS to existing system
Component C	Data Access Component	WS to existing system
Component D	Paper Digitising	COTS for scanning
Component E	Paper Digitising	Additional module to D
Component G	Paper Digitising	Individual software
Component H	Paper Digitising	Individual software
Component I	Accounting	SAP module for accounting
Component J	Accounting	Additional module to I
Component K	DMS	COTS
Component L	DMS	COTS
Component N	DMS	Individual software
Component P	DMS	Additional module to K
Component R	DMS	COTS
Component S	DMS	WS to existing module
Component T	Accounting	External commercial WS
Component Y	Archiving	Archiving system COTS
Component Z	Archiving	Archiving system COTS

Table 6.7: Software Components Subject to Evaluation

To visualise both functionality and business process coverage of each candidate, process model is updated to include component/activity mappings (see figure A.1 in appendix A). Note that, by design, the MultiSel implementation requires every single business process and activity to be covered by at least one software component. Thus, as some activities do not require any software support, a dummy component is introduced which is mapped to all the business processes that do not require software support.

Candidates are usually rated based on data taken from specifications, empirical evaluations or estimations. To allow for a comparison of the results of different software selection methodologies, equal input data must be used. Therefore, all candidates will be assigned a-priori ratings for each of their category objectives. These ratings will serve as data basis for all further rating, weighting and ranking.

Depending on whether category values can be measured in “real units” (e.g. monetary units, time units or measurable resource consumption) or not, different methods will be used for value definition. If a category can be measured with a discrete number that relates to a real unit, candidates will be assigned their absolute value for this category. Otherwise, if a category is made up of intangible assets such as *usability*, an abstract scale of *points* that ranges from 0 to 10 will be applied for measuring. See appendix A for the complete set of all candidate characteristics values.

#### 6.1.4 Constraints

The decision situation requires several constraints to be defined:

- Component A, the existing scan software, cannot be used together with the

potential new scan software, component D.

- The two DMS applications component K and component L cannot be used together.
- Component E is an extension module required by component D. While component E cannot be used without this extension, the extension itself (component D) can be used without component E.
- Component J is a plug-in for component I and cannot be used without it.
- Component P is an important extension for component K. While using component P without component K is not possible, using component K without component P yields a reduction of usability by 45 points.
- Initial costs must be equal or less than 140.000.
- Maintenance costs must be equal or less than 20.000 per year.
- Setup must be completed within a maximum of 135 days.

## 6.2 Analytic Hierarchic Process

The AHP requires all decision criteria to be ordered in a hierarchy. Figure 6.2 shows a hierarchic representation of the selection criteria collected in section 6.1.2. All objectives have been assigned indices (such as “L2.1” for “Functional Criteria”) to facilitate legibility and quotability during subsequent processing.

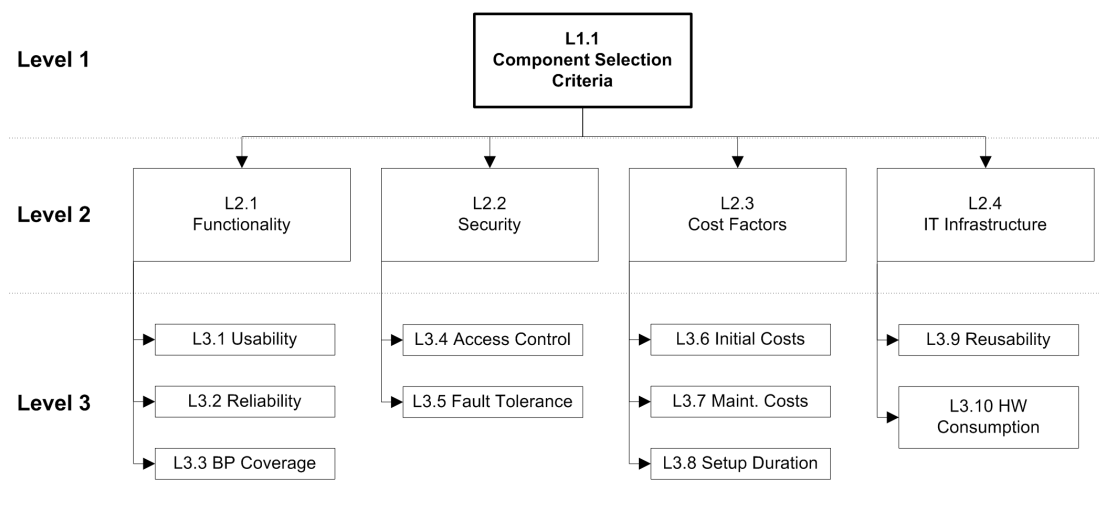


Figure 6.2: AHP Selection Criteria Hierarchy

### 6.2.1 Preconditions

When using the AHP, definition of parameters and weighting of objectives is usually performed by groups of multiple stakeholders. As simple discussion of objectives and finding arithmetic averages of user ratings is no relevant component of this thesis, group decision process will be treated as a “black box” delivering definite, discrete numeric values for all required parametrisations.

The comparison of criteria of different types, as pointed out in previous chapters, introduces the problem of having to express preference to one objective over another, which in certain cases can be a highly subjective decision. Because the generic AHP does not include functionality for mapping candidates to requirements (in our case, components to business processes), business process coverage is included as an objective of its own. It reflects the percentage of business processes covered by a single component.

The AHP technique currently applied does not support consideration of multiple periods, though evaluations could be performed for each period separately. In the following evaluations, this issue will be bypassed by calculation of period averages for all categories that have no constant value over all time periods. Another major drawback of the AHP is the inability of specifying the components’ respective functionality. This can be misleading as soon as two candidates subject to comparison belong to different application classes, because absolute values such as costs cannot be matched correctly. When, for example, costs of a “bigger” DMS are compared to a “smaller” web service extension, the latter candidate appears to be more cost efficient though it only offers a small part of the first candidate’s functionality. A possible approach of mitigating this fact consists in adding the number of each candidates’ supported business processes as an independent decision objective. Furthermore, after all candidates have been evaluated, a ranking is expected to exist that shows the potential overall value of all candidates but does not imply software combinations suited for achieving optimal business process coverage. Thus, a primitive knapsack algorithm will be conducted manually to find feasible combinations of candidates.

### 6.2.2 Calculating Objective Weights

In the first step, the AHP requires comparison of all objectives that are on a common hierarchy level and have a common parent objective. Objectives are compared to each other by pairwise comparison where preferences are expressed numerically using a scale from 9 (“much more important”) to 1 (“of equal importance”). Two compared objectives are always assigned reciprocal values, this means that if objective A is rated with 3, the rating for candidate B is implied as  $\frac{1}{3}$  (see table 6.8). After this, eigenvalues are calculated and the overall objective weight (column “Total”) is derived from the average of its row sum (see table 6.9).

This process is repeated for all objectives until each of them is assigned a total weight. Table 6.10 displays the total objective weights of the third level.

	L2.1	L2.2	L2.3	L2.4
L2.1	1,00	2,00	0,50	2,00
L2.2	0,50	1,00	2,00	3,00
L2.3	2,00	0,50	1,00	2,00
L2.4	0,50	0,33	0,50	1,00
SUM	4,00	3,83	4,00	8,00

Table 6.8: Comparison of Level 2 Objectives

	L2.1	L2.2	L2.3	L2.4	TOTAL
L2.1	0,25	0,52	0,13	0,25	0,29
L2.2	0,13	0,26	0,50	0,38	0,32
L2.3	0,50	0,13	0,25	0,25	0,28
L2.4	0,13	0,09	0,13	0,13	0,12

Table 6.9: Eigenvalues of Level 2 Objectives

### 6.2.3 Ranking Candidates

The next step appears problematic to perform, as the AHP requires all candidates to be compared to each other in respect to each of the lowest-level (L3) objectives. Thus, a total of 18 components must be juxtaposed and compared to each other which makes  $\frac{18^2-18}{2}$  comparisons. This process must be performed for 10 objectives resulting in a total of  $10 \frac{18^2-18}{2} = 1530$  single comparisons. Candidate ratings for each objective are presented in table 6.11, for details on comparison data, see appendix B.

To calculate overall ratings for candidates, all of the candidates' L3 objective values must be correlated (multiplied) with all of their parent objectives up to the root node and summed up (see equations 6.1 to 6.3).

$$Total_{(c,L3.1)} = Rating_{(c,L3.1)} Rating_{(L3.1)} Rating_{(L2.1)} Rating_{(L1.1)} \quad (6.1)$$

$$Total_{(c)} = \sum_{x=3;y=1}^{3;10} Rating_{(c,Lx.y)} Rating_{(L(x-1).(y-1))} \dots Rating_{(L1.1)} \quad (6.2)$$

$$Total_{(c_A,L3.1)} = 0.07 * 0.14 * 0.29 = 0.02842 \quad (6.3)$$

	L3.1	L3.2	L3.3	L3.4	L3.5	L3.6	L3.7	L3.8	L3.9	L3.10
VAL	0,14	0,24	0,62	0,67	0,33	0,23	0,54	0,16	0,25	0,75

Table 6.10: Total Weights of Level 3 Objectives



	L3.1	L3.2	L3.3	L3.4	L3.5	L3.6	L3.7	L3.8	L3.9	L3.10
A	3.00	0.67	1.97	0.38	1.03	3.44	3.52	3.15	2.83	2.93
B	1.51	1.26	0.32	2.07	2.96	1.85	1.28	0.71	3.84	5.33
C	3.00	0.67	0.32	2.33	1.84	1.84	2.44	0.16	3.84	3.85
D	1.51	1.67	1.97	2.07	1.84	2.19	2.44	2.85	4.63	2.93
E	0.72	0.32	0.32	0.18	0.24	3.78	4.44	3.52	1.11	2.49
G	0.72	1.26	3.96	1.21	3.33	3.07	3.52	3.15	2.83	3.85
H	0.30	1.67	0.32	2.33	1.84	1.84	2.44	2.53	2.83	2.93
I	0.30	1.26	3.96	2.33	3.33	1.70	4.44	3.89	6.22	1.71
J	2.41	1.67	1.97	2.33	4.56	3.44	4.67	4.22	0.28	0.31
K	0.72	2.67	3.96	1.21	1.84	0.65	0.96	1.42	3.84	4.93
L	1.51	0.32	3.96	0.71	0.24	2.19	2.44	2.53	1.11	2.93
N	1.51	2.67	5.22	3.22	3.33	1.28	1.28	2.48	5.22	2.93
P	2.41	1.67	1.97	2.33	3.33	3.44	4.67	4.22	1.86	0.31
R	1.51	1.26	5.22	1.21	0.55	2.19	0.55	3.52	0.28	4.93
S	2.41	2.67	1.97	2.33	1.84	3.44	4.67	4.44	0.28	0.31
T	4.67	2.67	1.97	2.33	1.03	3.44	4.67	4.22	1.11	0.31
Y	1.51	1.26	5.22	2.07	1.84	1.28	1.81	1.97	2.83	3.85
Z	2.41	2.67	5.22	2.33	2.96	0.16	0.16	0.71	5.22	5.33

Table 6.11: Ratings of Candidates for Level 3 Objectives

### 6.2.4 Composing a Portfolio

Final results of AHP evaluation are shown in table 6.12, candidates are ordered by descending overall value. As mentioned before, this list visualises all candidates' overall rankings but does not point out reasonable combinations of candidates. Thus, a primitive knapsack-like algorithm is applied<sup>2</sup>: Starting on top of the list in table 6.12, each candidate is tested regarding its business process coverage and its compatibility to all constraints defined in section 6.1.4. If inclusion of a component violates any constraints, it is omitted. Otherwise, if the number of business processes newly covered by the component is higher than the number of business processes it double-covers<sup>3</sup>, it is included.

Starting out from the first candidate, component N, we examine the candidate list. Candidate N has the highest ranking and covers four uncovered business processes, thus it is included in the portfolio. The next candidate, candidate I, covers one business process that is already covered and two that are not. It is included. Two of the four business processes covered by component Z are already covered by other components. Thus, it is not included in the portfolio. Candidate J covers two business processes and can only be included in the portfolio if candidate I is present too. As candidate I is already included in the portfolio, candidate J can be included either. Candidate G covers three business processes and thus is included. Now, only one business process is left to be covered, this can be done by either using component B or H. Because component B is ranked

<sup>2</sup>Of course, this algorithm could be optimised in many ways. It was kept simple intentionally as the AHP itself is primary subject of this evaluation.

<sup>3</sup>Covering a business process that is already covered by a previously evaluated component

higher than component H, component B is included in the portfolio.

	RATING	Relevance for Portfolio
N	0.0760	Included
I	0.0682	Included
Z	0.0662	Omitted (double-coverages)
J	0.0642	Included
G	0.0629	Included
P	0.0622	Omitted (double-coverages)
T	0.0617	Omitted (double-coverages)
S	0.0594	Omitted (double-coverages)
Y	0.0585	Omitted (double-coverages)
D	0.0525	Omitted (double-coverages)
K	0.0522	Omitted (double-coverages)
B	0.0481	Included
H	0.0480	Not tested (portfolio already complete)
R	0.0476	Not tested (portfolio already complete)
L	0.0471	Not tested (portfolio already complete)
C	0.0471	Not tested (portfolio already complete)
A	0.0431	Not tested (portfolio already complete)
E	0.0343	Not tested (portfolio already complete)

Table 6.12: Overall Candidate Ratings, Selected Candidates

Now the portfolio is complete as all business processes are covered. Table 6.13 lists the portfolio's category values as well as the category limits defined in section 6.1.2.5. As we can see, the portfolio obeys all limits thus being a valid solution. If one or more of the portfolio's category values had exceeded the previously defined limits, the process of portfolio composition would have been repeated selecting other candidates until a valid solution had been found.

	AC	FT	HW	IC	MC	RL	RU	SD	US
B	7	7	7	28	6	7	6	70	6
G	6	8	5	15	2	7	5	20	5
I	8	8	2	35	1	7	9	10	4
J	8	9	1	5	0	8	2	5	7
N	9	8	4	45	6	9	8	35	6
<b>Total</b>	<b>7.6</b>	<b>8</b>	<b>19</b>	<b>128</b>	<b>15</b>	<b>7.6</b>	<b>6</b>	<b>140</b>	<b>5.6</b>
Limit	-	-	-	140	20	-	-	135	-

Table 6.13: Components in the Resulting Portfolio

Of course, there are many more approaches for combining candidates and composing portfolios. While some of them might involve methodological adaptations (such as circumventing the issue of portfolios not regarding category limits by checking the portfolio category totals before adding a new component), others might stress more traditional “trial-and-error” techniques.

As there is a large number of possible solutions for this problem, finding an “ideal” portfolio can require significant efforts. As pointed out earlier, searching for an optimal solution by generation of various portfolios basing on the AHP output data lies beyond the scope of this thesis, as the result of the AHP itself and not the subsequent AHP portfolio composition algorithm is subject to examination.

## 6.3 Weighted Scoring Method

Weighted scoring methods essentially attempt to determine total weights for all candidates subject to evaluation by aggregating all of their objectives in a single total candidate weight [18]. This requires an explicit, declarative a-priori weighting of all the candidates objectives (cf. [38]).

### 6.3.1 Preconditions

To permit an aggregation of multiple objectives (categories), category values have to be balanced. This means that, for objective weighting to be effective and for the final output to be significant, all values must be in a common scope. When, for example, aggregating *initial costs* and *usability*, directly weighting both objectives is not effective as the range of initial costs usually exceeds the range of usability (which, in our decision situation, is limited to a maximum of 10) by far. Therefore, all objective values are normalised to fit in an integer scale ranging from 0 to 10. This is done using the equation shown in figure 6.4.

$$BalancedValue_{(a)} = \frac{ScaleMax = 10}{MAX_{i=1}^A(OriginalValue_{(i)})} OriginalValue_{(a)} \quad (6.4)$$

Moreover, as weighted scoring methods - in analogy to the AHP - do not produce whole software portfolios but only candidate rankings, the knapsack algorithm previously used in section 6.2.4 will be applied for portfolio composition.

### 6.3.2 Ranking Candidates

Weighted scoring method requires weights to be assigned to all categories (see table 6.14). By that, candidates' category values can be aggregated and summed up to a total weight for each candidate.

After category values have been normalised they can be aggregated. This is done by multiplying each of the candidates' category values with the corresponding category weight and summing up all of these weighted category values for each candidate (see figure 6.5).

$$TotalWeight_{(c)} = \sum_{i=1}^A Rating_{(i)} Weight_{(c,i)} \quad (6.5)$$

The final ranking of all candidates regarding their total weights is shown in table 6.15. It is notable that 100% of all candidates lie in 20% of the scale

SHORT	DESCRIPTION	UNIT	WEIGHT
AC	Access Control	Pts.	8%
BC	BP Coverage	No. of BPs	13%
FT	Fault Tolerance	Pts.	11%
HW	H/W Consumption	CPUs	8%
IC	Initial Costs	1.000 EUR	10%
MC	Maintenance Costs	1.000 EUR	13%
RL	Reliability	Pts.	11%
RU	Reusability	Pts.	11%
SD	Setup Duration	Working Days	7%
US	Usability	Pts.	8%
<b>Total</b>			<b>100%</b>

Table 6.14: Category Weights

	AC	BP	FT	HW	IC	MC	RL	RU	SD	US	Total
<b>P</b>	0.64	0.26	0.88	0.08	0.94	1.30	0.88	0.44	0.66	0.56	<b>6.64</b>
<b>J</b>	0.64	0.26	0.99	0.08	0.94	1.30	0.88	0.22	0.66	0.56	<b>6.53</b>
<b>I</b>	0.64	0.39	0.88	0.16	0.59	1.17	0.77	0.99	0.61	0.32	<b>6.52</b>
<b>T</b>	0.64	0.26	0.55	0.08	0.92	1.30	0.99	0.33	0.66	0.72	<b>6.45</b>
<b>S</b>	0.64	0.26	0.66	0.08	0.95	1.30	0.99	0.22	0.67	0.56	<b>6.33</b>
<b>G</b>	0.48	0.39	0.88	0.40	0.82	1.04	0.77	0.55	0.53	0.40	<b>6.26</b>
<b>D</b>	0.56	0.26	0.66	0.32	0.71	1.13	0.88	0.77	0.48	0.48	<b>6.25</b>
<b>N</b>	0.72	0.52	0.88	0.32	0.47	0.52	0.99	0.88	0.39	0.48	<b>6.17</b>
<b>A</b>	0.32	0.26	0.55	0.32	0.88	1.04	0.66	0.55	0.53	0.64	<b>5.75</b>
<b>H</b>	0.64	0.13	0.66	0.32	0.65	0.78	0.88	0.55	0.44	0.32	<b>5.37</b>
<b>Y</b>	0.56	0.52	0.66	0.40	0.47	0.65	0.77	0.55	0.31	0.48	<b>5.37</b>
<b>C</b>	0.64	0.13	0.66	0.40	0.65	0.78	0.66	0.66	0.00	0.64	<b>5.22</b>
<b>B</b>	0.56	0.13	0.77	0.56	0.67	0.52	0.77	0.66	0.09	0.48	<b>5.21</b>
<b>Z</b>	0.64	0.52	0.77	0.56	0.00	0.13	0.99	0.88	0.09	0.56	<b>5.14</b>
<b>E</b>	0.24	0.13	0.33	0.24	0.98	1.17	0.55	0.33	0.57	0.40	<b>4.94</b>
<b>R</b>	0.48	0.52	0.44	0.48	0.71	0.26	0.77	0.22	0.57	0.48	<b>4.93</b>
<b>K</b>	0.48	0.39	0.66	0.48	0.18	0.39	0.99	0.66	0.18	0.40	<b>4.81</b>
<b>L</b>	0.40	0.39	0.33	0.32	0.71	0.78	0.55	0.33	0.44	0.48	<b>4.73</b>

Table 6.15: Category and Total Weights of All Candidate

applied, namely between 4.65 and 6.65 on a scale that ranges from 0 to 10. Due to this, candidate ranking is very narrow and overall result is of low significance. This might occur because - disregarding category weighting which only has a certain influence on overall outcome - candidates' category sums are quite similar. Obviously, candidates yield too few difference for this approach to return a valid result.

### 6.3.3 Composing a Portfolio

Though validity of the obtained result appears doubtful, a software portfolio will be composed using the knapsack algorithm known from AHP post processing. Again, we start with the component of highest rank. Component P can only be included in a portfolio if K is too. As K is one of the lowest ranked components, P is omitted. Components J and I are included (J depends on I and I can be included too), T are omitted as its business process coverage is adequate to that of J. Component S is omitted too because it does not cover any uncovered processes, component G is included. Component D does not cover any new business processes and is omitted, component N is included, A is omitted and H is included. The final solution depicted in table 6.16 contains a valid portfolio.

	AC	FT	HW	IC	MC	RL	RU	SD	US
G	6	8	5	15	2	7	5	20	5
H	8	6	4	30	4	8	5	30	4
I	8	8	2	35	1	7	9	10	4
J	8	9	1	5	0	8	2	5	7
N	9	8	4	45	6	9	8	35	6
<b>Total</b>	<b>7.8</b>	<b>7.8</b>	<b>16</b>	<b>130</b>	<b>13</b>	<b>7.8</b>	<b>5.8</b>	<b>100</b>	<b>5.2</b>
Limit	-	-	-	140	20	-	-	135	-

Table 6.16: Components of Resulting Portfolio

## 6.4 Multiobjective Portfolio Approach

In analogy to the concept outlined in chapter 4, all required decision situation data is collected and entered into the multiobjective portfolio decision support application (in our case, the MultiSel implementation). This process is described in the following sections.

Data collection and input can be performed straight forward. Categories are entered together with their respective limits as defined in section 6.1.2.5. Components and business processes together with their mappings are imported from an ADONIS model. Thus, only required effort consists of entering candidate category values into the respective GUI and defining constraints. For all phases of data collection, screen shots of the MultiSel GUI showing the respective decision situation data can be found in appendix C.

### 6.4.1 Collection of Candidate Data

Category values of all candidates are obtained through research where possible. While hard fact such as costs or setup duration can be collected from manufacturer specifications easily, other facts such as the candidate's usability must be tested and estimated as outlined in section 4.2.3.

Because MultiSel permits definition of scales with arbitrary precision, and because all candidates' characteristics regarding the defined evaluation categories have been collected in a previous step (see appendix A), candidate data can be directly adopted and entered into the application. The scale that candidate values currently are applied to ranges from 0 to 10. To allow for a more precise calculation, all category values on this scale (values for absolute categories such as costs will not be changed) will be modified and represented on a scale that ranges from 0 to 100. This is accomplished by simple multiplication of all candidate category values by 10.

### 6.4.2 Modelling Constraints

All constraints defined in section 6.1.4 are categorised and modelled using one or more of the basic dependencies and relations introduced in section 4.2. Our first constraint, for example, states that candidate A cannot be used together with candidate D. Therefore, an exclusion constraint must be defined that limits all portfolios to include either candidate A or candidate D, but not both of them. Table 6.17 shows how this constraint is formulated.

Type	Description	Count	Components
Exclusion Constraint	Components A and D not together	1	A, D

Table 6.17: Constraint Example

As another example, candidate J cannot be included without candidate I as it strongly depends on it. Though, candidate I can be used without candidate J. A constraint must be defined that drops all portfolios which contain candidate J but not candidate I. As, in its current maturity stage, MultiSel does not natively permit formulation of "x not without y" constraints, this constraint must be implemented using a workaround: At first, an AND relation is defined which, for all portfolios that include component J, significantly increases a specific category value for a certain time period resulting in the respective portfolios' being dropped (due to the category value's exceeding a limit defined for this specific category and period). Then, another AND relation is defined that becomes active when a portfolio contains both candidates I and J decreasing the previously increased category value to normal.

Type	Description	Count	Components	Category	Mod.
AND Relation	I requires J (1)	1	J	Initial Cost	+100
AND Relation	I requires J (2)	2	I,J	Initial Cost	-100

Table 6.18: Relation Example

This simple trick results in the following evaluation behaviour: If candidate J is contained in a portfolio, one of the portfolio's category values increases over the category maximum which, at the end of this portfolio's evaluation, leads to the portfolio being dropped. Only if this portfolio also contains component I, both components are present which results in a reduction of the category's value to normal thus preventing it from being dropped. Table 6.18 shows how these relations are implemented. Appendix C shows a screen shot of the application where all constraints have been defined.

### 6.4.3 Evaluation

After all data has been entered into MultiSel and automatic evaluation has been performed, all feasible non-dominated solutions (software portfolios) are stored in the MultiSel database. In total, evaluation of this decision situation yields 51 efficient candidate combinations that satisfy all constraints. This solution data is analysed in analogy to [59] with a MODS analysis tool that was originally implemented for breakdown of R&D project portfolios.

Using this tool, category limits will be refined until just a small number of portfolios is left that fit the corporate needs best. Figure 6.3 shows the main screen of the analysis tool. By moving the red upper and lower rulers, category values can be limited reducing the number of possible solutions. Categories that keep their value constant over all periods (such as access control) can be limited by only limiting values of the first period as all portfolios that do not satisfy this limitations are dropped and there is no point in limiting each of this category's time periods. For categories that have values changing over time, limits can be adjusted for each time period to reflect specific requirements. In our case, narrowing down of results is performed as follows:

- At first, the limit for hardware consumption is set to a maximum of 20 CPUs which reduces the number of portfolios to 41 (see figure 6.4).
- After this, the minimum requirement for access control is set to 65 points (see figure 6.5).
- Then, maximum setup duration is set to 120 days (see figure 6.6).
- For usability, a minimum of 60 points is defined (see figure 6.7).
- Finally, portfolios are limited to have a minimal reusability of 45 points (see figure 6.8).

After the last limit has been defined, only 6 portfolios are left. Note that now, because the number of remaining portfolios has dropped below 10, each portfolio is represented by a separate bar. This renders final portfolio selection more transparent and gives an impression of each single portfolio's approximate characteristics. Now, when comparing portfolios regarding reusability (figure 6.8) as well as all other categories (displayed in figure 6.9), the third portfolio from the left (portfolio number 3) appears to be a good choice as it yields many benefits at relatively low costs.

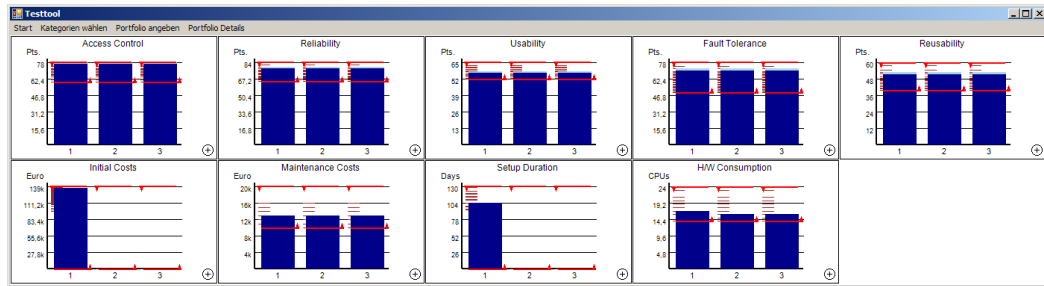


Figure 6.3: Main Screen of Analysis Tool

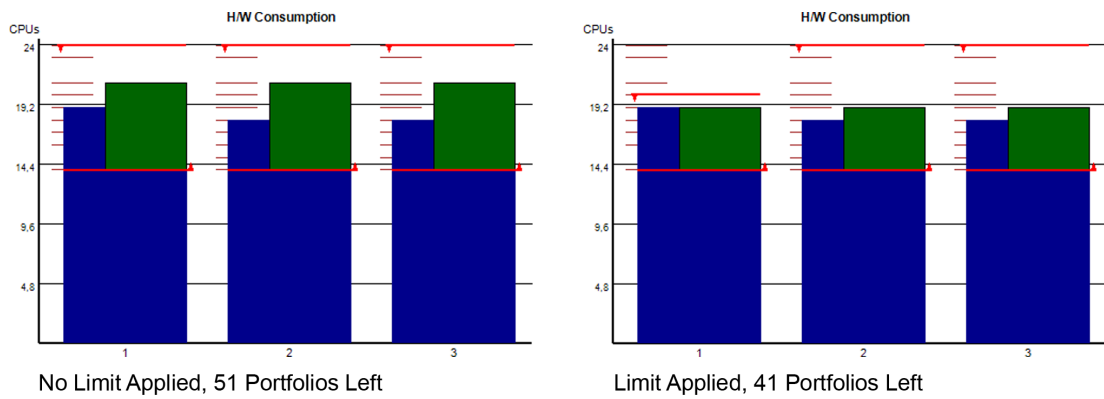


Figure 6.4: Restriction of Maximum Hardware Consumption

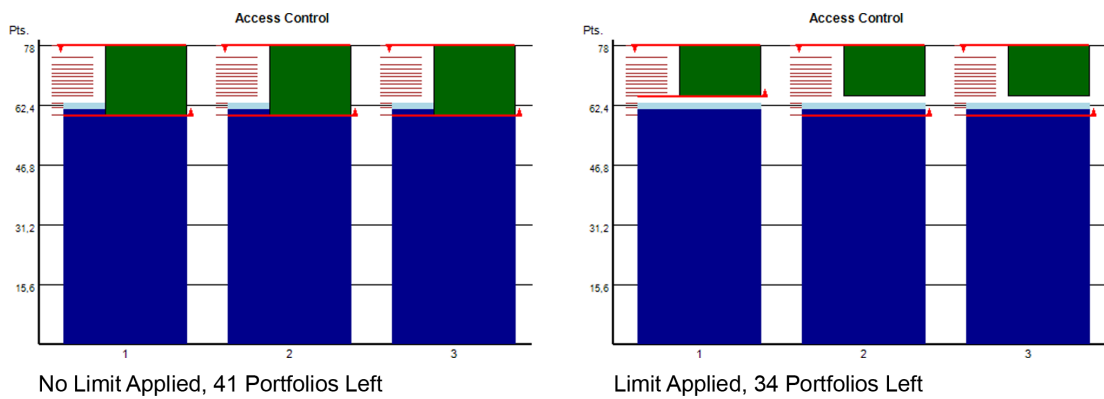


Figure 6.5: Restriction of Minimum Access Control



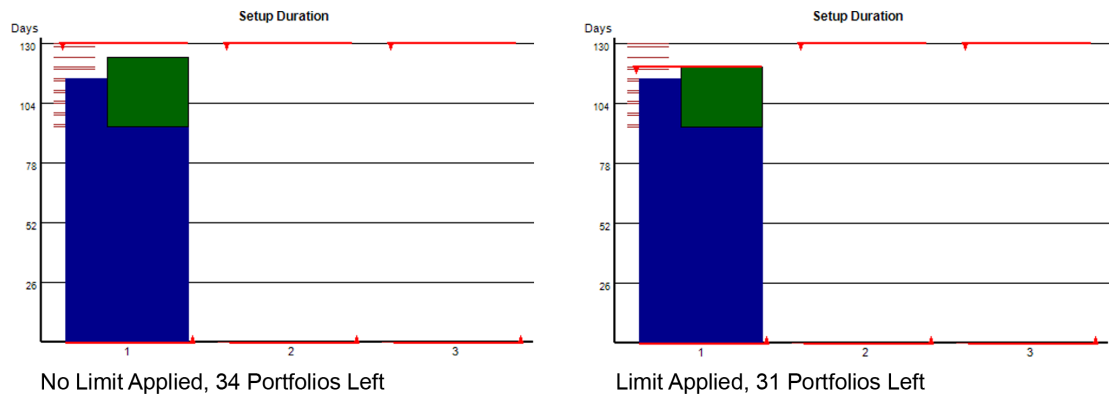


Figure 6.6: mRestriction of Maximum Setup Duration

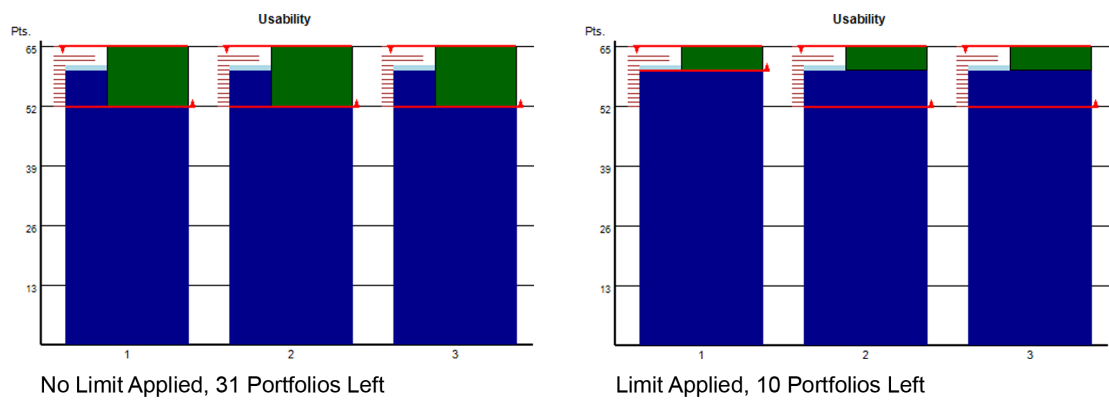


Figure 6.7: Restriction of Minimum Usability

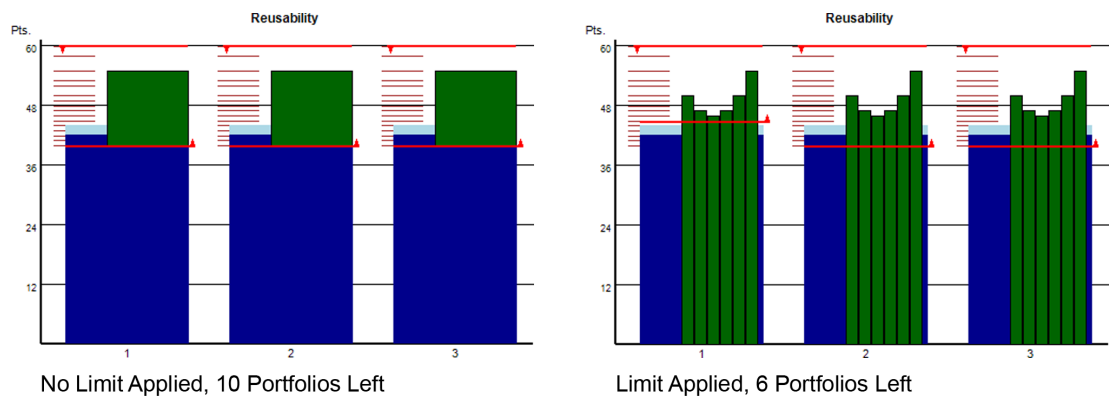


Figure 6.8: Restriction of Minimum Reusability

Of all remaining portfolios, portfolios number 3 exposes the highest values for access control, reliability and fault tolerance as well as the lowest initial costs, setup duration and hardware consumption. Its usability is quite high, and maintenance costs are the lowest of all portfolios during the first time period. Depending on whether or not this portfolio's poor reusability appears feasible, it can either be selected as final choice or evaluation can be continued by picking other portfolios and/or (re)defining additional category limits.

Because in this situation, the resulting reusability of around 4.6 points still appears high enough to provide sustainability of investments, portfolio number 3 will be adopted. Table 6.19 shows the portfolio's exact values.

As the MODS approach only yields solutions that satisfy all of the constraints and relations defined, it is evident that this portfolio is both efficient within solution space and valid as solution of this decision situation.

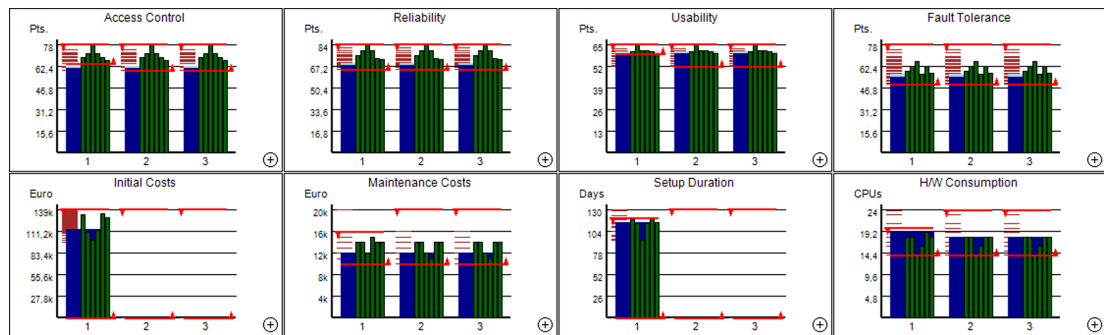


Figure 6.9: Category Values of Remaining Portfolios

	AC	FT	HW	IC	MC	RL	RU	SD	US
G	6	8	5	15	2	7	5	20	5
H	8	6	4	30	4	8	5	30	4
N	9	8	4	45	6	9	8	35	6
S	8	6	0	4	0	9	2	4	7
T	8	5	0	7	0	9	3	5	9
<b>Total</b>	<b>7.8</b>	<b>6.6</b>	<b>15</b>	<b>101</b>	<b>12</b>	<b>8.4</b>	<b>4.6</b>	<b>94</b>	<b>6.2</b>
Limit	-	-	-	140	20	-	-	135	-

Table 6.19: Components of Resulting Portfolio

## 6.5 Comparison of Results

When comparing results of algorithms which employ very different techniques to come to solutions of types that strongly differ from each other, it can be very difficult to make assertions regarding relative solution quality.

In this case study, two decision support methods that are commonly used for selecting *single* software components (AHP, WSM) have been compared to a newer approach that focuses on selection of whole sets of software component portfolios (MODS).

### 6.5.1 Significance

Though both AHP and WSM focus on finding one-dimensional ratings of components using relative weights (such as their value, quality or performance), an attempt was made to use these algorithms for composition of portfolios. To accomplish this, an additional algorithm had to be applied to the algorithms' outputs.

There is a nearly unlimited range of methods that can be applied to form portfolios out of ranked candidate lists, and developing an optimal algorithm to perform this task surely is subject to extensive evaluation and research. Thus, the portfolio forming algorithm used in this case study was kept simple to put focus on the respective base algorithms' output rather than the portfolio forming add-on algorithm. To preserve transparency and comparability, identical portfolio forming algorithms were applied to both AHP and WSM.

It is evident that even small behavioural changes in the applied portfolio forming algorithm could have resulted in completely different portfolios. Therefore, it is important to recall that solutions of AHP and WSM are exemplary for how portfolios can be derived using these methods, but they give no general indication on the relative quality of possible results.

### 6.5.2 Analysis of Results

Table 6.20 shows category totals of resulting portfolios, best values in each category are printed in bold. Due to organisations' individual demands regarding software portfolio characteristics it cannot be clearly determined which of the portfolios performs "best". Nevertheless it is not only evident that the solution generated with the MODS method appears absolutely feasible, but also that, if decision makers had aimed for a portfolio with different characteristics, the MODS method would have enabled them to efficiently explore solution space and find appropriate portfolios with little effort.

	AC	FT	HW	IC	MC	RL	RU	SD	US
AHP	7.6	<b>8.0</b>	19	128	15	7.6	<b>6.0</b>	140	5.6
WSM	<b>7.8</b>	7.8	16	130	13	7.8	5.8	100	5.2
MODS	<b>7.8</b>	6.6	<b>15</b>	<b>101</b>	<b>12</b>	<b>8.4</b>	4.6	<b>94</b>	<b>6.2</b>
Limit	-	-	-	140	20	-	-	135	-

Table 6.20: Results of Tested Software Selection Methods

For selection of single software components, WSM seems least-suited as in our case its results lack statistic significance. For finding software portfolios, the MODS approach appears very reasonable because from all solutions theoretically possible it preserves those that are acceptable within defined parameters and allows a transparent and straight-forward evaluation.

Because the MultiSel application returns *all* efficient (non-dominated) portfolios satisfying all constraints, it can be inductively stated that an arbitrary portfolio  $P$  is efficient if (and only if) it is an element of the MultiSel solution space. On the other hand, if portfolio  $P$  is *no* element of the MultiSel solution

space, it can be extrapolated that this portfolio has been dropped because either a) it is not dominated by at least one other portfolio or b) it does not satisfy all decision situation constraints.

The portfolio constructed with WSM is contained in the MultiSel solution space, which implies that the portfolio is Pareto efficient. The AHP portfolio, however, is not contained in the MultiSel solution space. As this portfolio was composed out of the AHP ranking manually respecting all constraints and inter-candidate dependencies, we can conclude that this portfolio has been dropped by MultiSel because it is not efficient. This does not necessarily mean that the AHP portfolio is dominated by the MODS solution portfolio. Much more, this means that the AHP portfolio is dominated by any of the portfolios in the MODS solution space.

The reason why the WSM returned an efficient solution while the AHP did not cannot be clearly isolated. But, due to the fact that none of these two algorithms considers the aspect of Pareto efficiency in any way, it appears to be sole coincidence driven by the algorithms' capability of finding adequate candidate rankings.

### 6.5.3 Method Characteristics and Behaviour

An important aspect that must be considered when comparing approaches is the way that solutions are achieved. For example, unlike the MODS approach, WSM and AHP require a-priori definition of all input parameters not permitting a-posteriori redefinition of preferences.

It is obvious that many advantages of the MODS approach in this context result from its portfolio-awareness. AHP and WSM for example cannot handle constraints and relations because such constructs just are not applicable within their models: The two methods do not aim at combining candidates and, by that, experience no need to consider total category sums or inter-candidate dependencies. Table 6.21 contains a summary of relevant aspects of software portfolio composition and information on how they are taken into account by each of the three selection approaches.

While AHP and WSM require the formulation of preferences as discrete values prior to evaluating candidates, the MODS approach permits a dynamic a-posteriori induction of preferences. Because the effect of changes in preference are visible in real-time, users can approximate to optimal solutions easily and in little time. In difference to AHP and WSM, MODS does not need direct comparison of category values of unequal types. By that, no bias is induced and evaluation process is kept transparent and simple. MODS rigorously eliminates dominated (non-efficient) solutions. This causes a significant reduction of the number of possible solutions. Though AHP and WSM natively do neither support candidate portfolios nor the paradigm of Pareto dominance, candidate efficiency is implicitly respected: Single candidates that are dominated by others have a lower over-all value and thus are ranked worse than their (more) efficient pendants.

Because AHP and WSM do not support candidate portfolios, their models cannot include inter-candidate dependencies or total value constraints. Thus, these

ASPECT	WSM	AHP	MODS
Definition of preferences	A-priori only, no a-posteriori definition of preferences possible.		A-priori definition of candidate characteristics and optional constraints, all further refinement is done a-posteriori.
Treatment of categories	Linear weighting of categories.	Hierarchic weighting of all categories.	No weighting of categories, step-wise a-posteriori exploration of solution space.
Elimination of dominated solutions	By ranking	By ranking	Yes
Consideration of inter-candidate dependencies	N/A	N/A	Yes
Consideration of value constraints	N/A	N/A	Yes
Consideration of BP coverage	No	No	Yes
Construction of portfolios	No construction of portfolios, evaluation of single candidates only.		Construction of portfolios only, no ranking of single candidates.

Table 6.21: Aspects of Software Portfolio Selection and Coverage

restrictions must be considered when manually constructing a portfolio based on the respective algorithm's candidate ranking. Due to its specialisation on software evaluation, the MODS model developed in course of this thesis includes software-specific concepts such as mapping candidates to business processes, which of course neither AHP nor WSM can offer.

## Chapter 7

# Conclusions

For today's companies, employing (and prior to this, choosing) appropriate software is an important success factor. Though there are many popular IT management and IT business alignment approaches, hardly any of them emphasises the use of specific methods to perform selection of software components. Because IT management is striving to continuously monitor and improve organisational IT landscape, evaluation of software portfolio performance should be performed on a regular basis. This appears impossible for many companies as they have not established standardised, well-defined processes for software evaluation yet. Currently, several approaches are used for evaluation of business applications. Though methods such as the AHP permit structured definition of preferences and analysis of single software candidates, their efficiency ceases as today's business often requires multiple software components that show a high coupling. By that, methodologies that consider multiple candidates under respect of their mutual dependencies are demanded.

The MODS approach appears well-suited for both pre-investment decision support and evaluation of present organisational software portfolios as it yields several important advantages over conventional decision support techniques. Because it requires no a-priori weighting of objectives, decision development process is highly transparent and result is very low biased as preferences are induced after portfolios have been constructed. By matching characteristics of whole sets of candidates with corporate strategy, holistic IT management approaches are endorsed while efficiency and sustainability of investments are optimised.

The MODS implementation *MultiSel* developed in course of this thesis has proven that multiobjective analysis permits a transparent and efficient selection of software portfolios. Though the MultiSel implementation mainly serves as proof of concept, the case study conducted proves that it is capable of providing solutions for small and medium scale business process scenarios.

## Chapter 8

# Directions for Further Research

As application of multiobjective methods to software portfolio decision problems is a relatively new approach, both model and implementation should be subject to further research.

## 8.1 Software Selection Model

Though the model put up in this thesis emphasises the use of an adapted OTSO method for collection of candidate characteristics, a more structured and more formal approach could increase transparency and reusability of candidate category data. For example, techniques such as the AHP's comparison matrix method that derives candidate category ratings from  $n : n$  comparisons of all candidates appear suited for improving the quality of a-priori candidate category values.

The current MODS model uses  $1 : n$  relations for mapping business processes to software candidates which means that of the  $n$  candidates that support a single business process, at least one must be selected for the business process to be considered *covered*. This results in the model not natively respecting the constellation of one business process requiring coverage by e.g. two different classes of software components and, in the current model, is mitigated by splitting up business processes until every sub process is coverable with a single software candidate. Thus, a reasonable extension of the model would consist in integrating consideration of multiple types of software candidates required to cover a single business process. In addition to this, portfolios' degree of business process support could be included in calculations as a variable. This would allow limiting the degree of business process coverage to a required minimum (less than the current 100 percent).

For combining candidates' category values to calculate portfolio totals, currently the aggregate functions *SUM* (e.g. portfolio initial costs = sum of all candidates' initial costs) and *AVERAGE* (e.g. portfolio reusability = average of all candidates' reusability) are implemented. To consider characteristics such as setup duration, which could - when setting up components simultaneously - be measured as the maximum setup duration of a single component within a portfolio (critical path), the use of further aggregate functions (e.g. *MAX* for setup duration) could be integrated.

Finally, the general model of multiobjective-driven decision support for portfo-



lio selection is capable of analysing decision situations in many further IT-related domains. The model could, for example, easily be adapted for application to hardware infrastructure optimisation, ASP service selection or various business process optimisations.

## 8.2 Implementation

An important and critical aspect of all portfolio models requiring enumeration of the whole solution space is performance. Though the current implementation performs well within the parameters given in the case study's decision situation, more complex parameter sets with higher numbers of objects (such as candidates, categories, periods, business processes etc.) can boost computation effort to an extent that renders the whole decision situation "unsolvable". Thus, not only application of high-performance implementation techniques but also introduction of heuristic algorithms (such as proposed in [28] or [15]) are major requirements for construction of larger-scale MODS systems.

To facilitate reuse of collected candidate data and by that permit regular evaluations of companies' software portfolios, implementation of system to save, load and manage base data of previously analysed candidates appears necessary.

## Appendix A

# Candidate Characteristics

	AC	FT	HW	IC	MC	RL	RU	SD	US
Component A	4	5	4	10	2	6	5	20	8
Component B	7	7	7	28	6	7	6	70	6
Component C	8	6	5	30	4	6	6	80	8
Component D	7	6	4	25	4	8	7	25	6
Component E	3	3	3	2	1	5	3	15	5
Component G	6	8	5	15	2	7	5	20	5
Component H	8	6	4	30	4	8	5	30	4
Component I	8	8	2	35	1	7	9	10	4
Component J	8	9	1	5	0	8	2	5	7
Component K	6	6	6	70	7	9	6	60	5
Component L	5	3	4	25	4	5	3	30	6
Component N	9	8	4	45	6	9	8	35	6
Component P	8	8	1	5	0	8	4	5	7
Component R	6	4	6	25	8	7	2	15	6
Component S	8	6	0	4	0	9	2	4	7
Component T	8	5	1	7	0	9	3	5	9
Component Y	7	6	5	45	5	7	5	45	6
Component Z	8	7	7	85	9	9	8	70	7

Table A.1: Component Characteristics, Period 1

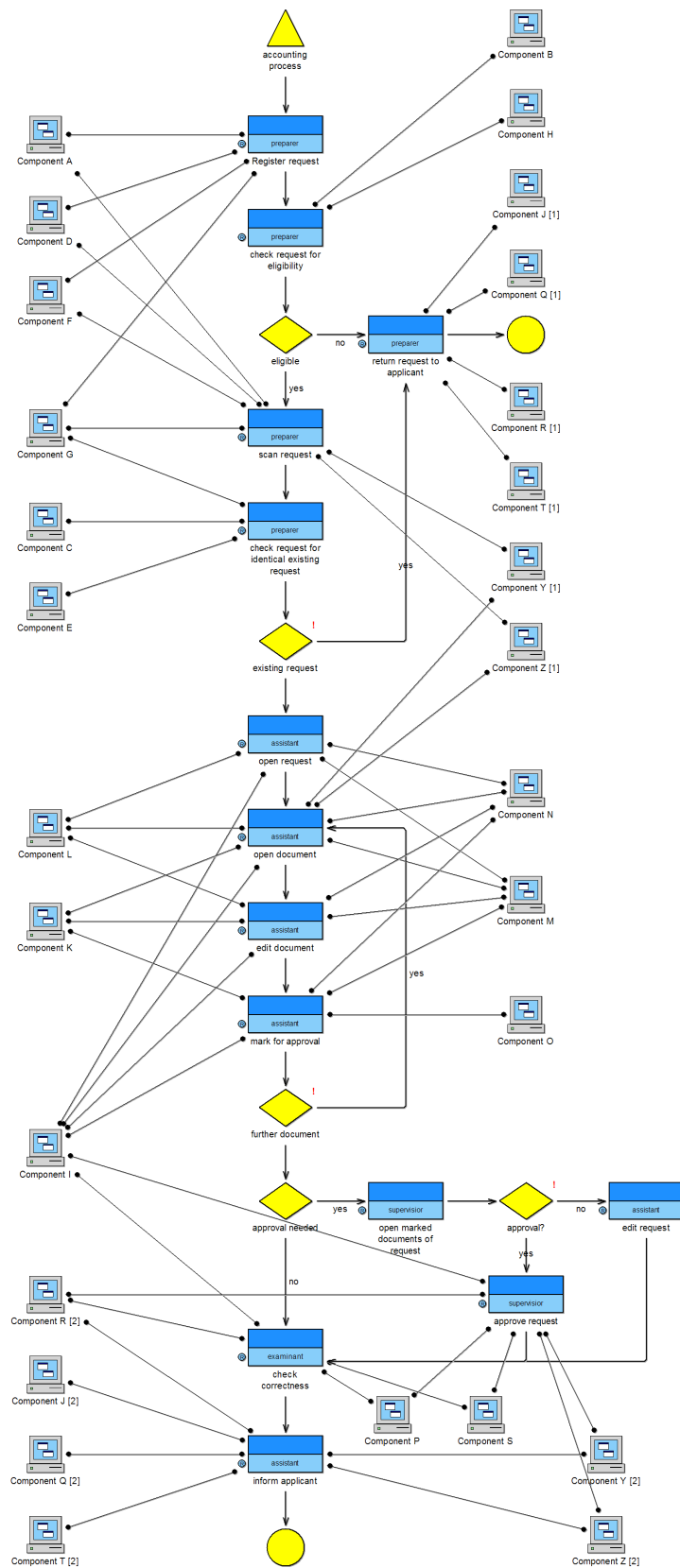


Figure A.1: BPM Including Candidate-Mappings

	AC	FT	HW	IC	MC	RL	RU	SD	US
Component A	4	5	4	0	2	6	5	0	8
Component B	7	7	7	0	6	7	6	0	6
Component C	8	6	5	0	4	6	6	0	8
Component D	7	6	4	0	0	8	7	0	6
Component E	3	3	3	0	1	5	3	0	5
Component G	6	8	5	0	2	7	5	0	5
Component H	8	6	4	0	4	8	5	0	4
Component I	8	8	2	0	1	7	9	0	4
Component J	8	9	0	0	0	8	2	0	7
Component K	6	6	6	0	7	9	6	0	5
Component L	5	3	4	0	4	5	3	0	6
Component N	9	8	4	0	6	9	8	0	6
Component P	8	8	1	0	0	8	4	0	7
Component R	6	4	6	0	8	7	2	0	6
Component S	8	6	0	0	0	9	2	0	7
Component T	8	5	1	0	0	9	3	0	9
Component Y	7	6	5	0	5	7	5	0	6
Component Z	8	7	7	0	9	9	8	0	7

Table A.2: Component Characteristics, Period 2

	AC	FT	HW	IC	MC	RL	RU	SD	US
Component A	4	5	4	0	2	6	5	0	8
Component B	7	7	7	0	6	7	6	0	6
Component C	8	6	5	0	4	6	6	0	8
Component D	7	6	4	0	0	8	7	0	6
Component E	3	3	3	0	1	5	3	0	5
Component G	6	8	5	0	2	7	5	0	5
Component H	8	6	4	0	4	8	5	0	4
Component I	8	8	2	0	1	7	9	0	4
Component J	8	9	0	0	0	8	2	0	7
Component K	6	6	6	0	7	9	6	0	5
Component L	5	3	4	0	4	5	3	0	6
Component N	9	8	4	0	6	9	8	0	6
Component P	8	8	1	0	0	8	4	0	7
Component R	6	4	6	0	8	7	2	0	6
Component S	8	6	0	0	0	9	2	0	7
Component T	8	5	1	0	0	9	3	0	9
Component Y	7	6	5	0	5	7	5	0	6
Component Z	8	7	7	0	9	9	8	0	7

Table A.3: Component Characteristics, Period 3

## Appendix B

# AHP Comparison Matrices

L2.1 - L2.4				
	L2.1	L2.2	L2.3	L2.4
L2.1	1	2	$\frac{1}{2}$	2
L2.2	$\frac{1}{2}$	1	2	3
L2.3	2	$\frac{1}{2}$	1	2
L2.4	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{2}$	1

Table B.1: AHP Comparison Matrix, Level 2.1 - 2.4

L3.1 - L3.3			
	L3.1	L3.2	L3.3
L3.1	1	$\frac{1}{2}$	$\frac{1}{4}$
L3.2	2	1	$\frac{1}{3}$
L3.3	4	3	1

Table B.2: AHP Comparison Matrix, Level 3.1 - 3.3

L3.4 - L3.5		
	L3.4	L3.5
L3.4	1	2
L3.5	$\frac{1}{2}$	1

Table B.3: AHP Comparison Matrix, Level 3.4 - 3.5

L3.6 - L3.8			
	L3.6	L3.7	L3.8
L3.6	1	$\frac{1}{2}$	2
L3.7	2	1	3
L3.8	$\frac{1}{2}$	$\frac{1}{3}$	1

Table B.4: AHP Comparison Matrix, Level 3.6 - 3.8

L3.9 - L3.10		
	L3.9	L3.10
L3.9	1	$\frac{1}{3}$
L3.10	3	1

Table B.5: AHP Comparison Matrix, Level 3.9 - 3.10

L3.1 (USABILITY)																			
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z	
A	1	3	1	3	5	5	7	7	1	5	3	3	1	3	1	1	3	1	
B	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
C	1	3	1	3	5	5	7	7	1	5	3	3	1	3	1	1	3	1	
D	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
E	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1	3	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	
G	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1	3	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	
H	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	9	$\frac{1}{5}$	$\frac{1}{5}$	
I	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	9	$\frac{1}{5}$	$\frac{1}{5}$	
J	1	3	1	3	3	3	5	5	1	3	3	3	1	3	1	$\frac{1}{3}$	3	1	
K	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1	3	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	
L	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
N	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
P	1	3	1	3	3	3	5	5	1	3	3	3	1	3	1	$\frac{1}{3}$	3	1	
R	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
S	1	3	1	3	3	3	5	5	1	3	3	3	1	3	1	$\frac{1}{3}$	3	1	
T	1	5	1	5	7	7	9	9	3	7	5	5	3	5	3	1	5	3	
Y	$\frac{1}{3}$	1	$\frac{1}{3}$	1	3	3	5	5	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	
Z	1	3	1	3	3	3	5	5	1	3	3	3	1	3	1	$\frac{1}{3}$	3	1	

Table B.6: AHP Comparison Matrix, Level 3.1

L3.2 (RELIABILITY)																			
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z	
A	1	$\frac{1}{3}$	1	$\frac{1}{5}$	3	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	3	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	
B	3	1	3	$\frac{1}{3}$	5	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	5	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
C	1	$\frac{1}{3}$	1	$\frac{1}{5}$	3	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	3	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	
D	5	3	5	1	7	3	1	3	1	$\frac{1}{3}$	7	$\frac{1}{3}$	1	3	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	
E	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	1	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{9}$	1	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	
G	3	1	3	$\frac{1}{3}$	5	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	5	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
H	5	3	5	1	7	3	1	3	1	$\frac{1}{3}$	7	$\frac{1}{3}$	1	3	$\frac{1}{7}$	$\frac{1}{3}$	3	$\frac{1}{7}$	
I	3	1	3	$\frac{1}{3}$	5	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	5	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
J	5	3	5	1	7	3	1	3	1	$\frac{1}{3}$	7	$\frac{1}{3}$	1	3	$\frac{1}{7}$	$\frac{1}{3}$	3	$\frac{1}{7}$	
K	7	3	7	3	9	3	3	3	3	1	9	1	3	3	1	1	3	1	
L	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	1	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{9}$	1	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	
N	7	3	7	3	9	3	3	3	3	1	9	1	3	3	1	1	3	1	
P	5	3	5	1	7	3	1	3	1	$\frac{1}{3}$	7	$\frac{1}{3}$	1	3	$\frac{1}{5}$	$\frac{1}{3}$	3	$\frac{1}{5}$	
R	3	1	3	$\frac{1}{3}$	5	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	5	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
S	7	3	7	3	9	3	3	3	3	1	9	1	3	3	1	1	3	1	
T	7	3	7	3	9	3	3	3	3	1	9	1	3	3	1	1	3	1	
Y	3	1	3	$\frac{1}{3}$	5	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	5	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
Z	7	3	7	3	9	3	3	3	3	1	9	1	3	3	1	1	3	1	

Table B.7: AHP Comparison Matrix, Level 3.2

L3.3 (BP COVERAGE)																			
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z	
A	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
B	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{7}$	1	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	
C	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{7}$	1	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	
D	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
E	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{7}$	1	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	
G	3	7	7	3	7	1	7	1	3	1	1	1	3	1	3	3	1	1	
H	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{7}$	1	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{3}$	
I	3	7	7	3	7	1	7	1	3	1	1	1	3	1	3	3	1	1	
J	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
K	3	7	7	3	7	1	7	1	3	1	1	1	3	1	3	3	1	1	
L	3	7	7	3	7	1	7	1	3	1	1	1	3	1	3	3	1	1	
N	3	9	9	3	9	1	9	1	3	1	1	1	3	1	3	3	1	1	
P	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
R	3	9	9	3	9	1	9	1	3	1	1	1	3	1	3	3	1	1	
S	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
T	1	3	3	1	3	$\frac{1}{3}$	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	
Y	3	9	9	3	9	1	9	1	3	1	1	1	3	1	3	3	1	1	
Z	3	9	9	3	9	1	9	1	3	1	1	1	3	1	3	3	1	1	

Table B.8: AHP Comparison Matrix, Level 3.3

L3.4 (ACCESS CONTROL)																			
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z	
A	1	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	3	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	
B	5	1	1	1	7	1	1	1	1	1	3	$\frac{1}{3}$	1	1	1	1	1	1	
C	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
D	5	1	1	1	7	1	1	1	1	1	3	$\frac{1}{3}$	1	1	1	1	1	1	
E	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	1	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	
G	3	1	$\frac{1}{3}$	1	5	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	
H	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
I	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
J	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
K	3	1	$\frac{1}{3}$	1	5	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
L	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	3	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{5}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	
N	7	3	1	3	9	3	1	1	1	3	5	1	1	3	1	1	3	1	
P	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
R	3	1	$\frac{1}{3}$	1	5	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	
S	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
T	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	
Y	5	1	1	1	7	1	1	1	1	1	3	$\frac{1}{3}$	1	1	1	1	1	1	
Z	5	1	1	1	7	3	1	1	1	3	3	1	1	3	1	1	1	1	

Table B.9: AHP Comparison Matrix, Level 3.4

L3.5 (FAULT TOLERANCE)																			
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z	
A	1	$\frac{1}{3}$	1	1	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	3	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	$\frac{1}{3}$	
B	3	1	1	1	7	1	1	1	$\frac{1}{3}$	1	7	1	1	5	1	3	1	1	
C	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
D	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
E	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{5}$	1	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	
G	3	1	3	3	7	1	3	1	1	3	7	1	1	5	3	3	3	1	
H	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
I	3	1	3	3	7	1	3	1	1	3	7	1	1	5	3	3	3	1	
J	5	3	3	3	9	1	3	1	1	3	9	1	1	7	3	5	3	3	
K	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
L	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{5}$	1	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	
N	3	1	3	3	7	1	3	1	1	3	7	1	1	5	3	3	3	1	
P	3	1	3	3	7	1	3	1	1	3	7	1	1	5	3	3	3	1	
R	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	3	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{3}$	3	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	
S	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
T	1	$\frac{1}{3}$	1	1	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	1	3	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	$\frac{1}{3}$	
Y	1	1	1	1	5	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5	$\frac{1}{3}$	$\frac{1}{3}$	3	1	1	1	1	
Z	3	1	1	1	7	1	1	1	$\frac{1}{3}$	1	7	1	1	5	1	3	1	1	

Table B.10: AHP Comparison Matrix, Level 3.5



L3.6 (INITIAL COSTS)																		
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z
A	1	1	3	1	1	1	3	3	1	5	1	3	1	1	1	1	3	7
B	1	1	1	1	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	3	1	3	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	3	5
C	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5
D	1	1	1	1	$\frac{1}{3}$	1	1	1	1	3	1	3	1	1	$\frac{1}{3}$	1	3	5
E	1	3	3	3	1	1	3	3	1	5	3	5	1	3	1	1	5	9
G	1	1	1	1	1	1	1	3	1	5	1	3	1	1	1	1	3	7
H	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	5
I	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	3	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	3
J	1	3	3	1	1	1	3	3	1	5	1	3	1	1	1	1	3	9
K	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	1
L	1	1	1	1	$\frac{1}{3}$	1	1	1	1	3	1	3	1	1	$\frac{1}{3}$	1	3	5
N	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	3
P	1	3	3	1	1	1	3	3	1	5	1	3	1	1	1	1	3	9
R	1	1	1	1	$\frac{1}{3}$	1	1	1	1	3	1	3	1	1	$\frac{1}{3}$	1	3	5
S	1	3	3	1	1	1	3	3	1	5	1	5	1	1	1	1	5	9
T	1	3	3	1	1	1	3	3	1	5	1	3	1	1	1	1	3	9
Y	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	1	3
Z	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{9}$	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	1

Table B.11: AHP Comparison Matrix, Level 3.6

L3.7 (MAINTENANCE COSTS)																		
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z
A	1	1	1	1	1	1	1	1	1	3	1	1	1	3	1	1	1	7
B	1	1	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	3
C	1	1	1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	5
D	1	1	1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	5
E	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
G	1	1	1	1	1	1	1	1	1	3	1	1	1	3	1	1	1	7
H	1	1	1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	5
I	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
J	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
K	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	3
L	1	1	1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	5
N	1	1	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	3
P	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
R	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{3}$	1	$\frac{1}{5}$	1	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{3}$	1
S	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
T	1	3	1	1	1	1	1	1	1	3	1	3	1	5	1	1	1	9
Y	1	1	1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	5
Z	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{9}$	1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{5}$	1

Table B.12: AHP Comparison Matrix, Level 3.7

L3.8 (SETUP DURATION)																		
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z
A	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
B	$\frac{1}{5}$	1	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	3
C	$\frac{1}{9}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{5}$	1
D	1	3	7	1	1	1	1	1	1	3	1	1	1	1	1	1	1	7
E	1	3	7	1	1	1	1	1	1	3	1	1	1	1	1	1	3	7
G	1	3	7	1	1	1	1	1	1	3	1	1	1	1	1	1	1	7
H	1	3	5	1	1	1	1	1	1	3	1	1	1	1	1	1	1	5
I	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
J	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
K	$\frac{1}{5}$	1	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	1	3
L	1	3	5	1	1	1	1	1	1	3	1	1	1	1	1	1	1	5
N	1	3	5	1	1	1	1	1	1	3	1	1	1	1	1	1	1	5
P	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
R	1	3	7	1	1	1	1	1	1	3	1	1	1	1	1	1	3	7
S	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
T	1	5	9	1	1	1	1	1	1	5	1	1	1	1	1	1	3	9
Y	$\frac{1}{3}$	1	5	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	5
Z	$\frac{1}{9}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{5}$	1

Table B.13: AHP Comparison Matrix, Level 3.8

L3.9 (REUSABILITY)																		
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z
A	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	$\frac{1}{3}$	1	5	5	3	1	$\frac{1}{3}$
B	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	1	3	5	5	3	1	1
C	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	1	3	5	5	3	1	1
D	1	1	1	1	5	1	1	1	7	1	5	1	3	7	7	5	1	1
E	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	3	$\frac{1}{3}$	1	$\frac{1}{5}$	1	3	3	1	$\frac{1}{3}$	$\frac{1}{5}$
G	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	$\frac{1}{3}$	1	5	5	3	1	$\frac{1}{3}$
H	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	$\frac{1}{3}$	1	5	5	3	1	$\frac{1}{3}$
I	3	3	3	1	5	3	3	1	9	3	5	1	3	9	9	5	3	1
J	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$
K	1	1	1	1	3	1	1	$\frac{1}{5}$	5	1	3	1	3	5	5	3	1	1
L	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	3	$\frac{1}{3}$	1	$\frac{1}{5}$	1	3	3	1	$\frac{1}{3}$	$\frac{1}{5}$
N	3	1	1	1	5	3	3	1	7	1	5	1	3	7	7	5	3	1
P	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	1	1	1	$\frac{1}{9}$	3	$\frac{1}{3}$	1	$\frac{1}{7}$	1	3	3	1	1	$\frac{1}{3}$
R	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	1	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{3}$	1	1	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$
S	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{7}$	1	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{9}$	3	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{3}$	1	3	3	1	$\frac{1}{5}$
T	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	3	1	1	$\frac{1}{3}$	5	1	3	1	3	1	3	1	$\frac{1}{3}$	$\frac{1}{5}$
Y	1	1	1	1	3	1	1	$\frac{1}{3}$	5	1	3	1	1	5	5	3	1	$\frac{1}{3}$
Z	3	1	1	1	5	3	3	1	7	1	5	1	3	7	7	5	3	1

Table B.14: AHP Comparison Matrix, Level 3.9

L3.10 (HW CONSUMPTION)																		
	A	B	C	D	E	G	H	I	J	K	L	N	P	R	S	T	Y	Z
A	1	$\frac{1}{3}$	1	1	1	1	1	3	5	1	1	1	5	1	5	5	1	$\frac{1}{3}$
B	3	1	1	3	3	1	3	5	9	1	3	3	9	1	9	9	1	1
C	1	1	1	1	1	1	1	3	7	1	1	1	7	1	7	7	1	1
D	1	$\frac{1}{3}$	1	1	1	1	1	3	5	1	1	1	5	1	5	5	1	$\frac{1}{3}$
E	1	$\frac{1}{3}$	1	1	1	1	1	1	3	$\frac{1}{3}$	1	1	3	$\frac{1}{3}$	3	3	1	$\frac{1}{3}$
G	1	1	1	1	1	1	1	3	7	1	1	1	7	1	7	7	1	1
H	1	$\frac{1}{3}$	1	1	1	1	1	3	5	1	1	1	5	1	5	5	1	$\frac{1}{3}$
I	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$	1	3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	3	$\frac{1}{3}$	3	3	$\frac{1}{3}$	$\frac{1}{5}$
J	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	1	1	$\frac{1}{7}$	$\frac{1}{9}$
K	1	1	1	1	3	1	1	3	7	1	1	1	7	1	7	7	1	1
L	1	$\frac{1}{3}$	1	1	1	1	1	3	5	1	1	1	5	1	5	5	1	$\frac{1}{3}$
N	1	$\frac{1}{3}$	1	1	1	1	1	3	5	1	1	1	5	1	5	5	1	$\frac{1}{3}$
P	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	1	1	$\frac{1}{7}$	$\frac{1}{9}$
R	1	1	1	1	3	1	1	3	7	1	1	1	7	1	7	7	1	1
S	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	1	1	$\frac{1}{7}$	$\frac{1}{9}$
T	$\frac{1}{5}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{5}$	1	$\frac{1}{7}$	1	1	$\frac{1}{7}$	$\frac{1}{9}$
Y	1	1	1	1	1	1	1	3	7	1	1	1	7	1	7	7	1	1
Z	3	1	1	3	3	1	3	5	9	1	3	3	9	1	9	9	1	1

Table B.15: AHP Comparison Matrix, Level 3.10

## Appendix C

# MultiSel Program Masks

☒ 1. Sessions
 ☐ 2. Categories
 ☐ 3. Components
 ☐ 4. Constraints/Relations
 ☐ 5. Business Processes
 ☐ 6. Start Evaluation

Active session: 8 Health Insurance

Active Session:

Sessions

ID	Description	Periods	Created	
1	Test1	3	23.01.2007 05:23	
2	Test2	10	23.01.2007 05:24	
3	Field Evaluation	8	23.01.2007 05:24	
4	Unnamed Decision Situation	6	23.01.2007 07:58	
5	Model Modification 1	4	30.01.2007 05:38	
6	Real Test Situation 1	4	31.01.2007 09:29	
7	ERP Selection	5	02.02.2007 12:35	
8	Health Insurance	3	02.02.2007 12:35	

[Add Session](#)

Figure C.1: MultiSel *Sessions* Mask

☐ 1. Sessions
 ☒ 2. Categories
 ☐ 3. Components
 ☐ 4. Constraints/Relations
 ☐ 5. Business Processes
 ☐ 6. Start Evaluation

Active session: 8 Health Insurance

Categories

ID	Description	Type	Optimize	Unit	Multi	Analysis	
23	Access Control	Benefit	<input type="checkbox"/>	Pts.	1	Average	
24	Reliability	Benefit	<input checked="" type="checkbox"/>	Pts.	1	Average	
25	Usability	Benefit	<input type="checkbox"/>	Pts.	1	Average	
26	Fault Tolerance	Benefit	<input checked="" type="checkbox"/>	Pts.	1	Average	
27	Initial Costs	Resource	<input checked="" type="checkbox"/>	Euro	1.000	Total	
28	Maintenance Costs	Resource	<input checked="" type="checkbox"/>	Euro	1.000	Total	
29	Setup Duration	Resource	<input checked="" type="checkbox"/>	Days	1	Total	
32	H/W Consumption	Resource	<input checked="" type="checkbox"/>	CPUs	1	Total	
33	Reusability	Benefit	<input checked="" type="checkbox"/>	Pts.	1	Average	

[Add Category](#)

Figure C.2: MultiSel *Categories* Mask

☐ 1. Sessions
 ☐ 2. Categories
 ☒ 3. Components
 ☐ 4. Constraints/Relations
 ☐ 5. Business Processes
 ☐ 6. Start Evaluation

**Active session: 8 Health Insurance**

**Components**

ID	Description		
111	Component A		
112	Component B		
113	Component C		
114	Component D		
115	Component E		
117	Component G		
118	Component H		
119	Component I		
120	Component J		
121	Component K		
122	Component L		
124	Component N		
126	Component P		
128	Component R		
129	Component S		
130	Component T		
131	Component Z		
132	Component Y		
133	Dummy Component		

[Add Component](#)

Import Components from Adonis XML file:  [Durchsuchen...](#) [Start Import](#)

**Period Values for Component A**

ID	Category	P1	P2	P3	
23	Access Control	40	40	40	
24	Reliability	60	60	60	
25	Usability	80	80	80	
26	Fault Tolerance	50	50	50	
27	Initial Costs	10	0	0	
28	Maintenance Costs	2	2	2	
29	Setup Duration	20	0	0	
32	H/W Consumption	4	4	4	
33	Reusability	50	50	50	

Figure C.3: MultiSel *Components* Mask (Showing Details for Comp. A)

☐ 1. Sessions   ☐ 2. Categories   ☐ 3. Components   ☒ 4. Constraints/Relations   ☐ 5. Business Processes   ☐ 6. Start Evaluation

**Active session: 8 Health Insurance**

**Category Limits**  
 Portfolios are dropped if their *total* value for a category in a period lies below (benefits) or above (resources) the defined minimum/maximum. Note that currently, categories marked as *average* cannot be limited.

ID	Category	Type	P1	P2	P3	
27	Initial Costs	Maximum (Total)	140		1	
28	Maintenance Costs	Maximum (Total)	20	20	20	
29	Setup Duration	Maximum (Total)	135			
32	H/W Consumption	Maximum (Total)				

**Inclusion/Exclusion Constraints**  
 Portfolios are dropped if they contain more/less than a defined number out of a list of candidates.

ID	Description	Type	Count	Components	
30	K or L	Maximum	1	Component K, Component L	
31	A or D	Maximum	1	Component A, Component D	

[Add Constraint](#)

**AND/OR Relations**  
 Portfolio values are modified if they contain at least/no more than a defined number out of a list of candidates.

ID	Description	Type	Limit	Components	Category	P1	P2	P3	
39	P only if K (1)	Minimum	1	Component P	Initial Costs			100	
40	P only if K (2)	Minimum	2	Component K, Component P	Initial Costs			-100	
41	J only if I (1)	Minimum	1	Component J	Initial Costs			100	
42	J only if I (2)	Minimum	2	Component I, Component J	Initial Costs			-100	
43	D only if E (1)	Minimum	1	Component D	Initial Costs			100	
44	D only if E (2)	Minimum	2	Component D, Component E	Initial Costs			-100	
45	K less usable if not P (1)	Minimum	1	Component K	Usability	-45	-45	-45	
46	K less usable if not P (2)	Minimum	2	Component K, Component P	Usability	45	45	45	






[Add Relation](#)

Figure C.4: MultiSel *Constraints/Relations* Mask

☐ 1. Sessions
 ☐ 2. Categories
 ☐ 3. Components
 ☐ 4. Constraints/Relations
 ☒ 5. Business Processes
 ☐ 6. Start Evaluation

**Active session: 8 Health Insurance**


**Business Processes**

ID	Description	Comment	
100	1. Register request	Imported from Model "example_mod_bp_final"	 
101	2. Check request for eligibility	Imported from Model "example_mod_bp_final"	 
102	5. Check for identical request	Imported from Model "example_mod_bp_final"	 
103	4. Scan request	Imported from Model "example_mod_bp_final"	 
104	3. Return request to applicant	Imported from Model "example_mod_bp_final"	 
105	6. Open request	Imported from Model "example_mod_bp_final"	 
106	8. Edit document	Imported from Model "example_mod_bp_final"	 
107	9. Mark for approval	Imported from Model "example_mod_bp_final"	 
108	7. Open document	Imported from Model "example_mod_bp_final"	 
109	10. Open marked request documents	Imported from Model "example_mod_bp_final"	 
110	13. Check correctness	Imported from Model "example_mod_bp_final"	 
111	14. Inform applicant	Imported from Model "example_mod_bp_final"	 
112	11. Edit request	Imported from Model "example_mod_bp_final"	 
113	12. Approve request	Imported from Model "example_mod_bp_final"	 

[Add Business Process](#)

Import Business Processes From Adonis XML File:  [Durchsuchen...](#) [Start Import](#)

**Component/Business Process Mapping**

ID	Component	Covered Business Processes	
111	Component A	1. Register request, 4. Scan request	
112	Component B	2. Check request for eligibility	
113	Component C	5. Check for identical request	
114	Component D	1. Register request, 4. Scan request	
115	Component E	5. Check for identical request	
117	Component G	1. Register request, 4. Scan request, 5. Check for identical request	
118	Component H	2. Check request for eligibility	
119	Component I	12. Approve request, 13. Check correctness, 9. Mark for approval	
120	Component J	14. Inform applicant, 3. Return request to applicant	
121	Component K	7. Open document, 8. Edit document, 9. Mark for approval	
122	Component L	6. Open request, 7. Open document, 8. Edit document	
124	Component N	6. Open request, 7. Open document, 8. Edit document, 9. Mark for approval	
126	Component P	12. Approve request, 13. Check correctness	
128	Component R	12. Approve request, 13. Check correctness, 14. Inform applicant, 3. Return request to applicant	
129	Component S	12. Approve request, 13. Check correctness	
130	Component T	14. Inform applicant, 3. Return request to applicant	
131	Component Z	12. Approve request, 14. Inform applicant, 4. Scan request, 7. Open document	
132	Component Y	12. Approve request, 14. Inform applicant, 4. Scan request, 7. Open document	
133	Dummy Component	10. Open marked request documents, 11. Edit request	

Import Component/BP Mapping From Adonis XML File:  [Durchsuchen...](#) [Start Import](#)

Figure C.5: MultiSel *Business Processes* Mask

1. Sessions 2. Categories 3. Components 4. Constraints/Relations 5. Business Processes 6. Start Evaluation

**Active session: 8 Health Insurance**

Decision Situation solved successfully.

**Evaluation of Decision Situation**

The Decision Situation has been solved under use of the following parameters:

**Portfolios found: 57**

Candidates: 19  
Categories: 9  
Business Processes: 14  
Periods: 3

Eval Input File: C:\Dokumente und Einstellungen\LENOVO\ASPNET\Lokale Einstellungen\Temp\tmp266.tmp  
Eval Output File: C:\Dokumente und Einstellungen\LENOVO\ASPNET\Lokale Einstellungen\Temp\tmp267.tmp  
Calculation Time: 0,97 secs.

Processor Output: [Show](#)  
Analysis: [Begin Result Analysis](#)

[Start Evaluation](#)

Figure C.6: MultiSel *Evaluation* Mask





## List of Figures

2.1	Evolution of IBM Service Revenues [83] . . . . .	8
2.2	IT Expenditure of Western European Banks 1999-2004 [33] . . . .	9
3.1	The BSM equation [73] . . . . .	13
3.2	The six levers of financial and real options [41] . . . . .	14
3.3	Enablers of business-IT alignment [45] . . . . .	16
3.4	Inhibitors of business-IT alignment [45] . . . . .	16
3.5	Strategic Alignment Model of Business and IT [63] . . . . .	17
3.6	Trends in Information Systems [2] . . . . .	18
3.7	Aspects of IT Governance [46] . . . . .	19
3.8	Service-oriented architecture - Find/Bind/Invoke [16] . . . . .	21
3.9	The SOA Framework [16] . . . . .	22
3.10	Example of a Simple IT Project Portfolio Analysis [33] . . . . .	24
3.11	Example: AHP decision network [40] . . . . .	29
3.12	Example for Comparison of Portfolios with Two Parameters . . .	30
3.13	Restriction of solution space [76] . . . . .	32
3.14	Factors influencing selection of criteria [39] . . . . .	39
4.1	Adapted OTSO Criteria Collection Process . . . . .	46
4.2	BPM Containing Software-Process Mappings . . . . .	53
5.1	The complete MultiSel data model . . . . .	58
5.2	ANDONIS Business Process Model . . . . .	63
5.3	Naming of Multiple Software Instances in Common BPMs . . . .	64
6.1	BPM: Processing of Incoming Requests . . . . .	68
6.2	AHP Selection Criteria Hierarchy . . . . .	73
6.3	Main Screen of Analysis Tool . . . . .	83
6.4	Restriction of Maximum Hardware Consumption . . . . .	83
6.5	Restriction of Minimum Access Control . . . . .	83
6.6	m . . . . .	84
6.7	Restriction of Minimum Usability . . . . .	84
6.8	Restriction of Minimum Reusability . . . . .	84
6.9	Category Values of Remaining Portfolios . . . . .	85
A.1	BPM Including Candidate-Mappings . . . . .	94
C.1	MultiSel <i>Sessions</i> Mask . . . . .	103
C.2	MultiSel <i>Categories</i> Mask . . . . .	103
C.3	MultiSel <i>Components</i> Mask (Showing Details for Comp. A) . . .	104
C.4	MultiSel <i>Constraints/Relations</i> Mask . . . . .	105

---

C.5	MultiSel <i>Business Processes</i> Mask . . . . .	106
C.6	MultiSel <i>Evaluation</i> Mask . . . . .	107

## List of Tables

3.1	Example: Criteria Comparison Matrix [40] . . . . .	30
3.2	Comparison of applicable valuation methods [40] . . . . .	35
4.1	Example for resource/benefit categories . . . . .	48
4.2	Example for category data over four time periods . . . . .	50
4.3	Example for category data over four time periods (for two candidates)	50
4.4	Example for a simple software-process mapping (matrix) . . . . .	53
5.1	Objects in the MultiSel Data Model . . . . .	59
6.1	Activities Performed in Request Processing . . . . .	67
6.2	Criteria Set Derived from Application Requirements . . . . .	69
6.3	Criteria Set Derived from Design Requirements . . . . .	69
6.4	Criteria Set Derived from Project Plan . . . . .	70
6.5	Criteria Set Derived from Organisational Requirements . . . . .	70
6.6	Final Set of Selection Criteria . . . . .	71
6.7	Software Components Subject to Evaluation . . . . .	72
6.8	Comparison of Level 2 Objectives . . . . .	75
6.9	Eigenvalues of Level 2 Objectives . . . . .	75
6.10	Total Weights of Level 3 Objectives . . . . .	75
6.11	Ratings of Candidates for Level 3 Objectives . . . . .	76
6.12	Overall Candidate Ratings, Selected Candidates . . . . .	77
6.13	Components in the Resulting Portfolio . . . . .	77
6.14	Category Weights . . . . .	79
6.15	Category and Total Weights of All Candidate . . . . .	79
6.16	Components of Resulting Portfolio . . . . .	80
6.17	Constraint Example . . . . .	81
6.18	Relation Example . . . . .	81
6.19	Components of Resulting Portfolio . . . . .	85
6.20	Results of Tested Software Selection Methods . . . . .	86
6.21	Aspects of Software Portfolio Selection and Coverage . . . . .	88
A.1	Component Characteristics, Period 1 . . . . .	93
A.2	Component Characteristics, Period 2 . . . . .	95
A.3	Component Characteristics, Period 3 . . . . .	95
B.1	AHP Comparison Matrix, Level 2.1 - 2.4 . . . . .	96
B.2	AHP Comparison Matrix, Level 3.1 - 3.3 . . . . .	96
B.3	AHP Comparison Matrix, Level 3.4 - 3.5 . . . . .	96
B.4	AHP Comparison Matrix, Level 3.6 - 3.8 . . . . .	97
B.5	AHP Comparison Matrix, Level 3.9 - 3.10 . . . . .	97

---

B.6	AHP Comparison Matrix, Level 3.1 . . . . .	97
B.7	AHP Comparison Matrix, Level 3.2 . . . . .	98
B.8	AHP Comparison Matrix, Level 3.3 . . . . .	98
B.9	AHP Comparison Matrix, Level 3.4 . . . . .	99
B.10	AHP Comparison Matrix, Level 3.5 . . . . .	99
B.11	AHP Comparison Matrix, Level 3.6 . . . . .	100
B.12	AHP Comparison Matrix, Level 3.7 . . . . .	100
B.13	AHP Comparison Matrix, Level 3.8 . . . . .	101
B.14	AHP Comparison Matrix, Level 3.9 . . . . .	101
B.15	AHP Comparison Matrix, Level 3.10 . . . . .	102

# Bibliography

- [1] W. M. P. Van Der Aalst. Making work flow: On the application of petri nets to business process management. *Lecture Notes in Computer Science*, 2360, 2002.
- [2] W. M. P. Van Der Aalst, Arthur H.M. ter Hofstede, and Mathias Weske. Business process management: A survey. *BPM 2003*, 2003.
- [3] Markus Andlinger. *Modell zur Projektauswahl und Ressourcenallokation*, 3 1997.
- [4] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. *Second International Conference On Software Engineering*, 1979.
- [5] Edward H. Bowman and Gary T. Moskowitz. Real options analysis and strategic decision making. *Organization Science*, 12, 2001.
- [6] Xavier Burgués, Illa Xavier Franch, and Joan Antoni Pastor. Formalising erp selection criteria. *Proceedings of the Tenth International Workshop on Software Specification and Design (IWSSD'00)*, 2000.
- [7] E. Chabrow. It staffs lack financial chops for project analysis. *Information Week*, 932:20, 2003.
- [8] R. Clarke and K. Steven. Evaluation or justification? the application of cost/benefit analysis to computer matching schemes. *Proceedings of European Conference on Computer Systems*, 1997.
- [9] Thomas E. Curtin. Business-it alignment - understanding your position. 1999.
- [10] D. D. Daniel. Management information crisis. *Harvard Business Review*, 39, 1961.
- [11] Sasha Dekleva. Justifying investments in it. *Journal of Information Technology Management*, 16, 2005.
- [12] Jörg Desel. Teaching system modeling, simulation and validation. *Proceedings of the 2000 Winter Simulation Conference*, pages 1669–1675, 2000.
- [13] Merriam-Webster Online Dictionary. Merriam-webster online dictionary. Online, 11 2006.

- [14] A. Dixit and R. Pindyck. *Investment under uncertainty*. Princeton University Press, 1994.
- [15] Karl Doerner. Ant colony optimization in multiobjective portfolio selection. *4th MIC*, 2001.
- [16] Schahram Dustdar. Service-oriented computing and web services. Presentation, 2005.
- [17] D. Jack Elzinga, Tomas Horak, Chung-Yee Lee, and Charles Bruner. Business process management: Survey and methodology. *IEEE Transactions on Engineering Management*, 42, 1995.
- [18] Alex A. Freitas. A critical review of multi-objective optimization in data mining: A position paper. *ACM SIGKDD Explorations Newsletter*, 6, 2004.
- [19] S. B. Graves, J. L. Ringuest, and J. F. Bard. Recent developments in screening methods for nondominated solutions in multiobjective optimization. *Computer Operations Research*, 19, 1992.
- [20] D. Greer and G. Ruhe. Software release planning: An evolutionary and iterative approach. *Information & Software Technology*, 46, 2004.
- [21] Erik Guldentops. Control and governance maturity survey: Establishing a reference benchmark and a self-assessment tool. *Information Systems Control Journal*, 6, 2002.
- [22] Erik Guldentops and Steven DeHaes. Cobit 3rd edition usage survey: Growing acceptance of cobit. *Information Systems Control Journal*, 6, 2002.
- [23] M. Hammer and J. Champy. *Reengineering the corporation: A manifesto for Business Revolution*. HarperBusiness, 1993.
- [24] J. Henderson and N. Venkatraman. Strategic alignment: A model for organizational transformation via information technology. *Working Paper 3223-90, Sloan School of Management, MIT*, 1990.
- [25] J. C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *Information Systems Management*, 2003.
- [26] Petri Hilli, Maarit Kallio, and Markku Kallio. Real option analysis of a technology portfolio. *Review of Financial Economics*, 2006.
- [27] Robert Hirschfeld and Katsuya Kawamura. Dynamic service adaptation. *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops '04*, 2004.
- [28] Jeffrey Horn. A niched pareto genetic algorithm for multiobjective optimization. *IEEE*, 1994.

- [29] ITGI. It governance institute. Online.
- [30] ITIL. The itil definition site. Online, 2006.
- [31] M. G. Iyigün. A decision support system for r&d project selection and resource allocation under uncertainty. *Project Management Journal*, 24, 1993.
- [32] D. Karagiannis. Business process management systems. *SIGOIS Bulletin*, 1995.
- [33] Bert Kersten and Chris Verhoef. It portfolio management: A banker's perspective on it. *Cutter IT Journal*, 16, 2003.
- [34] Bert Kersten and Han Verniers. Investing in it portfolio management: Growth path to value management. *Submitted to International Journal for Project Management*, 2006.
- [35] Jack P.C. Kleijnen. Bayesian information economics: An evaluation. *Interfaces*, 1980.
- [36] Christopher Koch. A new blueprint for the enterprise. *CIO Magazine*, 3/05, 2005.
- [37] J. Kontio. Otso: A systematic process for reusable software component selection. *University of Maryland Technical Reports*, 1995.
- [38] Jyrki Kontio. A case study in applying a systematic method for cots selection. *Proceedings of ICSE-18*, 1996.
- [39] Jyrki Kontio, Gianluigi Caldiera, and Victor R. Basili. Defining factors, goals and criteria for reusable component evaluation. *CASCON '96*, 1996.
- [40] Vincent S. Lai, Robert P. Trueblood, and Bo K. Wong. Software selection: A case study of the application of the analytical hierarchical process to the selection of a multimedia authoring system. *Information & Management*, 36, 1999.
- [41] K.J. Leslie and M. P. Michaels. The real power of real options. *The McKinsey Quarterly*, 3, 2000.
- [42] Harold A. Linstone and Murray Turoff. *The Delphi Method - Techniques and Applications*. 2002.
- [43] J. N. Luftman, P. R. Lewis, and S. H. Oldach. Transforming the enterprise: The alignment of business and information technology strategies. *IBM Systems Journal*, 32, 1993.
- [44] Jerry Luftman. Assessing business/it alignment. *Information Systems Management*, 2003.
- [45] Jerry N. Luftman, Raymond Papp, and Tom Brier. Enablers and inhibitors of business-it alignment. *Communications of AIS*, 1, 1999.



- [46] Zach Luse. *COBIT 4.0: Control Objectives for Information and related Technology*. ISACA, 2006.
- [47] Neil A. Maiden and Cornelius Ncube. Acquiring cots software selection requirements. *IEEE Software*, 1998.
- [48] Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 1952.
- [49] Harry M. Markowitz. Foundations of portfolio theory. *Journal of Finance*, 46, 1991.
- [50] Jurgen Martens, Ferdi Put, and Etienne Kerre. A fuzzy set theoretic approach to validate simulation models. *ACM Transactions on Modeling and Computer Simulation*, 16, 2006.
- [51] Andrew McAfee. Do you have too much it? *MIT Sloan Management Review*, 45, 2004.
- [52] F. W. McFarlan. *Portfolio approach to information systems*, volume 59. 1981.
- [53] Rita G. McGrath. A real options logic for initiating technology positioning investments. *Academy of Management Review*, 22, 1997.
- [54] Microsoft. Windows life-cycle policy. Online, 4 2007.
- [55] Ignitia Motjolopane and Irwin Brown. Strategic business-it alignment, and factors of influence: A case study in a public tertiary education institution. *Proceedings of SAICSIT 2004*, 2004.
- [56] T. Neubauer, C. Stummer, and E. Weippl. Workshop-based multiobjective security safeguard selection. *IEEE Proceedings of the First International Conderence on Availability, Reliability and Security (ARES 2006)*, 2006.
- [57] Thomas Neubauer. Multiobjective decision support for the business process driven allocation of service oriented architectures. *Technical Report*, TR1-11, 2006.
- [58] Thomas Neubauer and Christian Stummer. Extending business process management to determine efficient it investments. *SAC '07*, 2007.
- [59] Thomas Neubauer and Christian Stummer. Interactive decision support for multiobjective cots selection. *HICSS '07*, 2007.
- [60] Jakob Nielsen. *Usability Engineering*. B&T, 1994.
- [61] Mark E. Nissen. Valuing it through virtual process measurement. *ICIS '94*, 1994.
- [62] R. Nolan and F. W. McFarlan. Information technology and the board of directors. *Harvard Business Review*, 2005.

- [63] Raymond Papp. Business-it alignment: productivity paradox payoff? *Industrial Management & Data Systems*, 8, 1999.
- [64] Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The capability maturity model: guidelines for improving the software process*. Addison-Wesley Longman Publishing Co., Inc., 1995.
- [65] Chris Peltz. Web services orchestration and choreography. *IEEE Computer*, 36, 2003.
- [66] Joe Peppard. Managing it as a portfolio of services. *European Management Journal*, 21, 2003.
- [67] Blaize Horner Reich and Izak Benbasat. Measuring the linkage between business and information technology objectives. *MIS Quarterly*, 20/1, 1996.
- [68] Blaize Horner Reich and Izak Benbasat. Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly*, 24/1, 2000.
- [69] J. F. Rockart. Chief executives define their own data needs. *Harvard Business Review*, 57, 1979.
- [70] Pablo Rossi and Zahir Tari. Software adaptation for service-oriented systems. *MW4SOC '06*, 2006.
- [71] T. L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, New York, 1980.
- [72] Peter G. Sassone. Cost benefit analysis of information systems: A survey of methodologies. *ACM*, 1988.
- [73] Roger Smith. Applying options theories to technology management decisions. *CTO Network*, 2004.
- [74] Charles R. Standridge. Teaching simulation using case studies. *Proceedings of the 2000 Winter Simulation Conference*, 2000.
- [75] R. E. Steuer, L. R. Gardiner, and J. Gray. A bibliographic survey of the activities and international nature of multiple criteria decision making. *Journal of Multi-Criteria Decision Analysis*, 1996.
- [76] Christian Stummer and Kurt Heidenberger. Interactive r&d portfolio analysis with project interdependencies and time profiles of multiple objectives. *IEEE*, 2003.
- [77] C. Symons. It governance framework. *Forrester Best Practices*, 2005.
- [78] Bernadette Szajna. Software evaluation and choice: Predictive validation of the technology acceptance instrument. *MIS Quarterly*, 18, 1994.
- [79] TechnologyEvaluation. Technologyevaluation website. Online, 2006.

- [80] Chris Tiernan and Joe Peppard. Information technology: Of value or a vulture? *European Management Journal*, 22:609–623, 2004.
- [81] A Min Tjoa and Dimitris Karagiannis. It governance - definition, standards & zertifizierung. *Praxis und Wissen*, 4, 2005.
- [82] C. Verhoef. Quantitative it portfolio management. *Science of Computer Programming*, 45, 2002.
- [83] Eric Viardot. Key features and importance of professional information technology-based services. *European Management Journal*, 18:454–461, 2000.
- [84] Alain Wegmann, Pavel Balabko, Lam-Son Lê, Gil Regev, and Irina Rychkova. A method and tool for business-it alignment in enterprise architecture. *Proceedings of the CAiSE'05 Forum*, 2005.
- [85] Rosnah Mohd Yusuff, Kok PohYee, and M.S.J. Hashmi. A preliminary study on the potential use of the analytical hierarchical process (ahp) to predict advanced manufacturing technology (amt) implementation. *Robotics and Computer Integrated Manufacturing*, 17, 2001.