Doctoral Thesis

Risk analysis of information systems by agent-based modeling of business processes

Author(s):
Kaegi, Manuel

Publication Date:
2009

Permanent Link:
https://doi.org/10.3929/ethz-a-006104436

Rights / License:
In Copyright - Non-Commercial Use Permitted
Risk Analysis of Information Systems by Agent-based Modeling of Business Processes

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Sciences

presented by
MANUEL KAEGI
Dipl. Phys. Uni. Bern
born 14 February 1975
citizen of Bauma - Switzerland

accepted on the recommendation of
Prof. Dr. W. Kröger, examiner
Prof. Dr. E. Zio, co-examiner
Dr.-Ing. R. Mock, co-examiner

2009
Acknowledgements

This thesis is the result of the research work I have carried out at the Laboratory for Safety Analysis (LSA) between 2004 and 2008. It is co-financed by the Swiss Innovation Promotion Agency (KTI/CTI) in cooperation with major Swiss enterprises.

First of all, I would like to thank my doctoral co-advisor, Dr.-Ing. Ralf Mock, for his great support. He was a perfect guide on the way trough my thesis and a very competent advisor during all my work. He was on my side whenever I faced new challenges during this work and his experience in many fields of science and application where always of great value. Above all it has to be mentioned, that Ralf continued supporting me for more than one year even after he left the LSA which I appreciated very much.

A special thank I would like to extend to my doctoral advisor, Prof. Dr. Wolfgang Kröger director of LSA, for having accepted, supported and lead my research work. I am very grateful for the valuable dialogues, careful work reviews as well as for providing the opportunity to work in a well organized and inspiring environment.

During my research activity at ETH Zurich, I enjoyed great support of many People to which I all owe a dept of gratitude. My very special thanks are extended to Roberto Nibali and Ruth Ziegler for the support and instructions in Java programming and a great collaboration within a KTI/CTI-Project. Further more, I’d like to thank Markus Schläpter for being a very nice fellow and roommate at LSA and of course for all the fruitful dialogues and shared projects. Moreover I would like to extend my thanks to Sigrid Wagner and Domenico Salvati for all the inspiring discussions and for their contribution to the very agreeable atmosphere at the LSA.

My Wife Lisa Morgenthaler and my colleges of basisnote AG I’d like to thank for their support and patience.
Abstract

Today, many business processes are based on information systems, i.e., on complex and dynamic networks composed of hardware and software. The failure-free performance of such systems is often essential for the successful business result of affected companies. Therefore, these systems must not fail. To ensure, that the right investments are made, it is recommended to evaluate costs and benefits of each action. Thereby, frequencies and consequences (risk) of failures should be minimized. Established risk analysis methods of information systems faces higher and higher challenges and requirements day by day. Common techniques are often applicable only to a limited extend.

This thesis was accomplished in the context of a Swiss CTI-project\(^1\) together with partners from the industry\(^2\), public authorities\(^3\) and academia\(^4\). The main goal of this project was to develop, test and implement a simulation based approach for information system risk analysis.

The proper identification of risks within an existing information system has to account for many influence factors. On the technical side, this comprises the consideration of the (complex) network architecture, the failure mechanisms of components (computer, printer, etc.) and the interdependencies among hardware and software among others. The established risk analysis methods like check-lists, audits or even simulations are not able to handle this complex situations.

The problem specifications on the part of the CTI-project includes to develop an approach for risk analysis of information systems, which is tailored to the needs of com-

---

\(^1\)CTI is the Swiss Confederation’s innovation promotion agency. It supports the knowledge and technology transfer between companies and universities by bringing them together as partners on applied research and development projects. (www.kti-cti.ch)

\(^2\)InfoGuard, cablecom, Zobrist AG

\(^3\)the governor’s office of Aargau, the Swiss Federal Strategy Unit for IT and the Swiss Federal Office for National Economic Supply

\(^4\)ETH Zurich / Laboratory for Safety Analysis, HTA Luzern / Institute for Secure Software
mon companies. Therefore, the analysis procedure should start by the capturing of all business processes linked to the corresponding information system. These non-technical processes are often well known within the companies and they affect directly their business goals. The idea is, to first transform the multilayered sequences and links in a sequence diagram, which shall serve as a basis for setting up of an agent-based simulation model.

Within this work, three key aspects of the development of the demanded simulation approach are documented: 1) The selection of a suitable business process modeling language and their extension for risk analysis purposes. 2) An extensive feasibility study to demonstrate the feasibility of agent-based modeling for risk analysis and 3) the developed meta-model and its programmed interface for performing computer based simulation. All three key subjects are complemented with case-studies to illustrate the feasibility.

From a scientific point of view, the middle part of this thesis is the most attractive one. The integration of agent-based modeling in risk analysis opens a wide spectrum for research and development. Within this thesis, it is demonstrated for instance, that agent-based models are applicable for the modeling of complex systems also in the field of risk analysis, although it is almost not used in this field. Especially the capability of reproducing (real) complex system behavior with an agent-based computer simulation is regarded as one of the major strengths of this approach. As this kind of simulations allow the generation of risk-numbers, e.g., frequencies and consequences with ease, they are regarded as particularly attractive in this field of application.

Throughout the development of the simulation approach for risk analysis of information systems, one important focus was to tailor the approach to the needs of business practitioners, i.e. possible future users of the compiled software. Thus, a tool has been developed which is able to allocate weaknesses and risks in the business processes of information systems and it highlights the potentials for process optimization. Thereby, it is accounted for the interdependencies among processes, physical networks and components including humans. An assortment of these "objects" is compiled in a software-library, which can be connected by drag-and-drop to the process model. A modular structure allows to increase the granularity of the models step by step. If the relevant parameters of all objects are set, the risk of information system failure is computed by simulation.

Case-studies and the collaboration with the partners of the CTI-project finally enabled the development of a promising methodology for the application e.g. in the field of companies security-audits.
Zusammenfassung


Die Vorgabe des KTI-Projekts war, einen neuartigen Ansatz der unternehmensnahen Risikoanalytik von Informationssystemen zu entwickeln: An dessen Anfang soll die Erfassung der Geschäftsprozesse stehen, die zu einem Informationssystem verknüpft.

---

5 KTI ist die Förderagentur für Innovation des Bundes und unterstützt insbesondere den Technologietransfer zwischen Unternehmen und Hochschulen. (www.kti-citi.ch)
6 InfoGuard, Cablecom, Zobrist AG
7 Staatskanzlei Aargau, Informatikstrategieorgan Bund, Bundesamt für wirtschaftliche Landesversorgung
8 ETH Zürich / Lab. für Sicherheitsanalytik, HTA Luzern / Institut für Sichere Softwaresysteme


Fallbeispiele und die Kooperation mit den KTI-Partnern führten zu einer viel versprechenden Methodik, die vor allem für Security-Audits von Unternehmen einsetzbar ist.
Contents

1 Introduction 1
   1.1 Background ............................................. 1
   1.2 Problem Description ...................................... 1
      1.2.1 Characteristics of Information Systems .......... 2
      1.2.2 Risk Analysis ................................... 3
      1.2.3 State of the Art ................................ 4
   1.3 Scientific Goals ....................................... 6
   1.4 Structure of the Thesis ............................... 7

2 Risk Analysis 9
   2.1 The Concept of Risk ..................................... 9
   2.2 Classical System Analysis Methods ................... 13
      2.2.1 Logic Tree Analysis .......................... 13
      2.2.2 Tabular-based Methods .................... 15
   2.3 Modeling and Simulation ............................. 16
      2.3.1 Random Number Models ....................... 18
      2.3.2 Discrete Event Models .................... 19
      2.3.3 Continuous Time Models .................... 20
      2.3.4 Risk Number Computation ................... 20
   2.4 Information Systems and Risk Analysis ........... 22
      2.4.1 Information System ............................ 22
      2.4.2 State of the Art in IS Risk Analysis ....... 22
### 3 Business Process Modeling for Risk Analysis

3.1 Business Processes ........................................... 25
   3.1.1 BP Modeling Techniques ............................... 25
   3.1.2 Low Acceptance of Petri Nets ......................... 26
   3.1.3 Specified Requirements ............................... 27
3.2 Introduction to BPMN ........................................ 29
   3.2.1 Basic BPMN Elements .................................. 29
3.3 Extending BPMN for Risk Analysis ......................... 32
3.4 Case Study ................................................... 37
3.5 Conclusions .................................................. 41

### 4 Agent-based Modeling for Risk Analysis

4.1 Agent-based Modeling ........................................ 43
   4.1.1 Terms and Definitions ................................. 44
   4.1.2 History of ABM ......................................... 45
   4.1.3 ABM in Risk Research ................................. 46
4.2 Open Questions ............................................... 47
4.3 Case Study 1: Epidemic Propagation of Diseases ............ 48
   4.3.1 Virus Propagation Model .............................. 49
   4.3.2 Simulation of an Epidemic ............................ 50
4.4 Case Study 2: Maintenance Processes ....................... 52
   4.4.1 Markov Model ........................................... 52
   4.4.2 Monte Carlo Simulation ............................... 54
   4.4.3 Cut-off Criteria ....................................... 56
   4.4.4 Agent-based Model .................................... 56
   4.4.5 Extending the Model .................................. 58
   4.4.6 Evaluating Maintenance Strategies .................. 60
   4.4.7 Optimal Strategies .................................... 60
   4.4.8 Emergent System Properties ......................... 61
A.1.1 Business process modelling using BPMN ........................... 120
A.2 com.xj.anylogic ......................................................... 122
A.2.1 Engine ................................................................. 122
A.2.2 ActiveObject ......................................................... 122
A.3 The Library Package (bpram) ........................................ 123
A.3.1 Activity ............................................................... 123
A.3.2 PhysicalObject ..................................................... 131
A.3.3 Events ................................................................. 132
A.3.4 Gateways ............................................................ 134
A.3.5 Entities ............................................................... 136
A.4 bpram.basics .......................................................... 138
A.4.1 ActiveObjectExtended ............................................. 138
A.4.2 CombineExtended ................................................ 139
A.4.3 ConcurrSubTokenHandler ...................................... 139
A.4.4 MatchQExtended .................................................. 139
A.4.5 StatsAnalyzer ....................................................... 140
A.4.6 SubActivity ......................................................... 140
A.4.7 SyncSubTokenHandler .......................................... 141
A.4.8 TokenDuplicator .................................................. 141
A.4.9 VirtualTokenRouter .............................................. 141

B Terminology .............................................................. 143
Chapter 1

Introduction

1.1 Background

The end of the last millennium was characterized by the emergence and success of the so called information society. This implicated the growth of information-based economic activities and enabled the emergence of novel industries, products, and services. This trend was driven by the increasing functionality and performance of information technology (IT) products, accompanied by continuously decreasing prices. The development of new communication technologies supported the cost efficient decentralization, cross-linking and sharing of IT infrastructures which finally resulted in a global network of interdependent systems and agents. As the key component of this global information infrastructure the Internet has emerged. It has become a basis for thousands of business applications and the main platform for information exchange for millions of people. However, all these modern comforts have come along with novel challenges and problems which have to be addressed by our society.

1.2 Problem Description

Most of the companies make an intense use of information and communication technology (ICT). The reliable operation of their information system (IS) is essential for the successful progress of the daily business. Therefore the high dependency on complex ICT demands an adequate treatment of the associated risks.

Topics such as network topology and stability for instance have been discussed extensively in the past [1]. Furthermore it is popular among scientists to ask global questions
like "What happens to our economy and society if the Internet breaks down?" or "Is it possible for a skilled hacker to control our critical infrastructures through the Internet like it was suggested by the recent Hollywood movie Die Hard 4.0?"

But in fact, there are a lot of unsolved problems on much lower scale, which have to be addressed as well. Till this day, the application of probabilistic risk analysis methods (PRA) to IS on the company level is not standard. Instead, most companies rely on simple checklists when evaluating the risk of their IS. The reason for this omission lies mainly in a lack of quantitative methods and standards tailored to the specific problems of middle scaled IS.

In the last decade, big companies, e.g., insurances, banks, and consultants have started to developed quantitative approaches for analyzing the risk of their own IS. These advances are often not public as the corresponding companies are not willing to share their achievements. As a consequence, these commercial approaches do not underly a scientific review process which promotes skepticism on the quality and correctness of the underlying methodology. For this reason, there is not only a scientific but also an economic motivation for more scientific efforts in this field.

### 1.2.1 Characteristics of Information Systems

ICT includes a multitude of technologies and services like telecommunication devices and networks, computer hardware and software. The following listing emphasizes major characteristics of IS:

- IS are generally highly **distributed**. There is often no one having administrative access to all system components. → In case of emergency, system operators from different units with different interests have to cooperate.

- High dependency on **process architecture**, i.e. the processes (e.g., control or maintenance) itself are influencing, e.g., the reliability and availability of a given system.

- IS are characterized by a very high **connectedness** meaning that it is in principle possible to access any IT device from a remote or home computer via physical networks like the Internet.

- Modern IS are subject to continuous and frequent changes.
1.2. Problem Description

As a result of the high complexity and dynamics, the modeling of these systems is a challenging task.

1.2.2 Risk Analysis

The basic objective of risk analysis and management is to secure service providing by technical systems and to ensure or increase their reliability and availability. The essential procedure of a technical risk analysis is well established and comprises the identification and estimation of potential hazards which threaten a system. Risk analysis is usually embedded in a risk management process (see Figure 1.1). As the concept of risk is well established in many industries, it is regarded as self-evident that it is also valuable for IS.

In general, there are two kinds of risk analysis techniques: The qualitative and the quantitative (probabilistic) approach. Qualitative approaches are very valuable in practice because they are simple to use and provide quick results. As a matter of fact, most companies judge risk analysis method by the presence or absence of a corresponding standard. On this account, standards as the best practice ISO/IEC-17799 [3] and the "BSI IT-Grundschutzhandbuch" [4] are very popular. These methods mainly rely on expert judgment which is appropriate for many systems, but has its limitations when the complexity is increasing. Above all, the scientific value of the development of new qualitative approaches is limited.

<table>
<thead>
<tr>
<th>Risk Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Assessment</td>
</tr>
<tr>
<td>Risk Analysis</td>
</tr>
<tr>
<td>Source Identification</td>
</tr>
<tr>
<td>Risk Estimation</td>
</tr>
<tr>
<td>Risk Evaluation</td>
</tr>
<tr>
<td>Risk Treatment</td>
</tr>
<tr>
<td>Risk Avoidance</td>
</tr>
<tr>
<td>Risk Optimization</td>
</tr>
<tr>
<td>Risk Transfer</td>
</tr>
<tr>
<td>Risk Retention</td>
</tr>
<tr>
<td>Risk Acceptance</td>
</tr>
<tr>
<td>Risk Communication</td>
</tr>
</tbody>
</table>

Figure 1.1: Risk management system [2]
In contrast, the quantitative or probabilistic risk analysis (PRA) of complex systems is a very challenging task. Up to now, estimating risk through PRA has primarily been a non behavioral, physical engineering approach to assessing the reliability of a system, running either in series or parallel or both, where PRA aims to quantify and rank the risks caused by technical installations. The ranking can be used to decide, how to allocate resources in an optimal way in order to reduce the overall risk. For companies with high dependency on complex IS, PRA is a very interesting approach to secure the system in an efficient way. But most PRA methods are time consuming and in addition, there exists no established PRA method or standard to quantify the risks of IS. Thus, the risk analysts have to put up with the question whether their toolbox, which is filled with established techniques (e.g., Failure Mode and Effects Analysis (FMEA) [5], Fault Tree Analysis (FTA) [6], Event Tree Analysis (ETA) [7], Markovian Chains [8]), is still sufficient for analyzing such systems. Considering the increasing risk caused by ICT, this situation is not satisfying. As long as there is no risk analysis method, which is able to tackle the dangers rising from ICT, it is important to promote the developments in this field of research.

1.2.3 State of the Art

In the past, a major concern in IS operation was the warranty of IT-security. This includes the analysis of the "CIA triad", i.e. the confidentiality, integrity and availability of a given system. As a consequence, many standards and regulations (e.g., [3, 4]) address security related matters. Regarding the reliability analysis of electronic components, the well established US military handbook MIL-HDBK-217F [9] and the Telcordia SR-332 (former Bellcore) still seem to be best practice [10].

In contrast to these established frameworks, risk analysis of IS is a relatively new discipline which has just began to gain momentum. Consequently, there is a multitude of questions and problems which needs to be addressed.

So far, many risk analysis techniques have been developed for hazard identification and risk evaluation purposes in other fields, e.g., in the chemical industry and nuclear power generation. These approaches are reflected in various standards (e.g., [11]) and have been successfully used to analyze and manage large-scaled closed systems for about 60 years. But at large, this kind of risk analysis techniques are considered as suitable to only a limited extent in IT. Where the qualitative approaches such as FMEA are generally not able to handle complex and distributed IS [12], the established PRA techniques like Markovian Chains or Fault Tree Analysis (FTA) have specific shortcom-
ings. Markov Chains suffer a inescapable state explosion when applying them to large systems. Fault Trees in fact are approved for the analysis of large systems such as nuclear power plants, but they come to their limits when the system models comprise of feedback loops. Petri nets [13] may overcome some of the theoretical problems but the computation of risk numbers (e.g., frequencies and consequences) with Petri nets has turned out to be inefficient [14–16].

Recent research mainstream has tended to focus on computer based simulation models such as Monte Carlo simulation or cellular automate for analyzing complex systems [17], but these approaches do not fully meet the business needs. Representers of Swiss companies such as swisscom, cablecom, SwissRe, UBS and InfoGuard have indicated in several workshops 
\(^1\), that there is a need for more practical approaches compatible with current business practices such as business process models.

In the past, several studies addressed these problems in IS risk analysis and it was tried to develop a number of tools and methods which were aimed to be business compatible and capable of representing complex IS. Examples for this achievements are, e.g., CRAMM [18, 19] or MARION [20]. Where MARION uses a scenario-based approach, the CRAMM methodology bases on a cardinal evaluation technique which originates in the US standard FIPS 65 [21]. Another attempt yielded the CORAS framework [22–24] which uses an object oriented approach by means of UML profile (Unified Modeling Language) for model-based risk analysis purposes, all these approaches lack of a probabilistic representation.

More recently, literature [25] has emerged that offers novel ideas on how to model IS for risk analysis, e.g., to model IS by event driven scenarios (EDS). Advanced modeling techniques (e.g., Petri nets and discrete event models) have been adopted for examining critical scenarios within complex IS. As a conclusion of these recent studies, the idea emerged to use the agent-based modeling (ABM) concept for the numerical analysis of IS.

During the last years, the number of research activities in the field of ABM has considerably increased. ABMs are regarded as a new approach enabling an important step forward in system modeling and simulation, technology and theory [26].

An ABM is a object-oriented model designed for computer simulation. The idea is to model only the components of a system (agents) by giving them rules of behavior. Then to start a computer simulation of all components in parallel in order to analyze the generation of macro structures, often referred to as emergent properties. ABM are

\(^1\)Workshops within the university-industry-consortium "Risk Management and Modeling for Distributed Systems", see http://www.risk-analysis.ch
well adaptable to complex systems with various types of components and multilayer structures such as IS. One example of an achievement in the field of ABM related to risk analysis is the study of Helbing [29]. He investigated a situation of escape panic in an overcrowded area with limited exits. Using an agent-based simulation model of human behavior, it was found that placing an obstacle in front of an emergency exit can increase the efficiency of such an exit due to unexpected self-organization. During the last years, other interesting risk related studies using agent-based models were done by Bonabeau [26,30], Shargel [31] Xie [32], Kerr [33] or Liao [34,35]. They are investigating selected emergent behaviors of complex systems but they don’t study or apply any concept of risk.

1.3 Scientific Goals

The main research goal of this thesis is the development of a new method for the quantitative risk analysis of IS. Thereto, it is intended to make use of ABM, one of the most advanced and modern modeling techniques. The choice of ABM is based on the fact, that it promises to cover the main requirements for modeling the complexity of IS. Especially the modeling of interdependencies among autonomous interacting elements (agents) is a strength of this approach. Further more, ABM includes hybrid modeling, i.e. the possibility of simultaneously modeling continuous time and discrete event systems. This is essential for describing a real world system. The integration of ABM in a risk analysis methodology is an innovation as till now there is only very few research activities in this field, although it leads to powerful modeling capabilities. In contrast to established modeling approaches in risk analysis, ABM is able to simulate emergent behavior of a given system. Emergent system behavior appears when a number of simple elements (agents) operate in an environment, forming a more complex and unexpected behavior as a collective. Especially if such emergent behavior is risk relevant, e.g. dangerous or expensive, the capability of detecting it is relevant. The challenge of this integration is to make no restriction in the modeling capabilities but simultaneously to offer a business compatible tool for the analysis of complex IS.

As specified above, it turned out as a result of several workshops with business representers, that business process modeling techniques should be taken into account, when developing a novel risk analysis method for IS. It was made the unanimous statement by these representers, that most of the problems rising from business IS don’t originate just from technical failures, but from ill-designed business processes. Often it is a combination of both, technical failures and process collapse. Accordingly, the
output of a risk analysis should also trace the weak points and bottlenecks of business processes and not only the vulnerabilities of the technical system. While in the past, the developments in business process management focused on the visualization of processes [36], one emphasis of this work is to develop methods for the quantification of the consequences of process innovations. Thereto, the concept of risk is regarded as adequate.

An IS, represented by a business process model, follows procedural steps from task commencement to its successful fulfillment. This is in contrast to risk analysis approaches which stress failures among the negative rated aspects. To achieve a convincing risk analysis method, both system modeling approaches have to be combined. For this purpose, a risk metric has to be developed, i.e. anticipate a combination of frequency and consequence, which enables the rating of each path (scenario) through a business process. To observe the behavior of a given IS over time and thus to quantify the frequencies of all possible scenarios, it is intended to run agent-based computer simulations of the modeled IS. Accordingly, the ABM serves as a scenario generator. The scientific prospect of such an extended simulation model is the possibility of detecting unknown (or unexpected) emergent phenomena.

To achieve this goal, it is proposed in this work to build up a computer-based software library which facilitates the modeling of interactions among risk-relevant component agents (computers, routers, servers, etc.) in an ABM and to link it with the corresponding IS, represented by business processes (process agents) and the involved personal (human agents) see Fig. 1.2. The basic model of an IS will consist of at least one business process (abstract agent) and an underlying technical system (physical agents).

As business processes are generally success oriented as mentioned above, they usually don’t address sub-processes such as failure elimination and debugging during the execution of a business process function which is casually done by the employees. Therefore, maintenance processes have to be implemented, e.g., in the agents’ behavior.

### 1.4 Structure of the Thesis

After this introduction, Chapter two gives an overview of classical risk analysis concepts and methods in general and points out the specific situation of risk analysis in connection with IS. The Chapter three introduces and discusses the application of business process modeling techniques to IS and points out, how they can be adapted for risk
analysis purposes. Chapter four presents in detail the agent-based modeling concept. To demonstrate the potentials of this approach relating to risk analysis, three case studies are presented there. Chapter five elaborates on the combination of agent-based and business process modeling techniques for the development of a novel framework for IS risk analysis. As a result, a generic JAVA based software library for the risk analysis of IS is introduced. This library is designed to convert the semi-formal business process model (BPM) of an IS into a formal simulation model. Chapter six demonstrates the capabilities with a case study of a large IS of the Swiss Federal Strategy Unit for IT, before concluding with discussion of results achieved and an outlook to future work.
Chapter 2

Risk Analysis

2.1 The Concept of Risk

The basic objective of risk management is to secure technical systems and to increase their reliability, availability and safety. In particular, the procedure of a technical risk analysis comprises the identification and the rating of potential hazards which threat a system. As a rating metric, a risk parameter $\mathcal{R}$ is defined (see section 2.1). The ranking of the risk-rated hazards derived by a risk analysis can be used to decide, how to allocate recourses in an optimal way in order to reduce the overall risk.

Def. 1 Risk ($\mathcal{R}$) is a "combination of the likelihood of an event and its consequence" [11], where frequency rather than probability may be used in describing likelihood.

Within this thesis, risk is expressed in mathematical terms as:

$$\mathcal{R} = f(frequency, consequence)$$

(2.1)

and is used as a rating metric of hazardous and undesired events or event sequences (subsequently called scenarios $s_i$).

Def. 2 Scenario ($s_i$) is an event or an event sequence of a specified system which causes, with a probability $p_i$, an undesired consequence $C_i$.

The likelihood of a scenario is indicated by the scenario frequency $f_{s,i}$ and the likelihood of a consequence triggered by a scenario $s_i$ is represented by the consequence frequency $f_{c,i} = p_i \cdot f_{s,i}$, where $p_i = Pr\{C_i|s_i\}$.
Thus risk $\mathcal{R}$ is to see always in context of a scenario $s_i$ which can cause an undesired outcome. This outcome may be varicolored and it is a challenge to scale it to a single unit and summarized it in a total consequence $C_i$. For instance, it is often desired to compute the possible consequence $C_i$ as a monetary value. Furthermore, the consequence $C_i$ of a specified scenario $s_i$ is not a priori a sharp value, but rather a range of values with an appropriate probability distribution and a mean value $\overline{C_i}$. For discrete distributions, $\overline{C_i}$ is defined as

$$
\overline{C_i} = \sum_{k=0}^{\infty} p_k \cdot k.
$$

(2.2)

Where $p_k = Pr\{C_i = k|s_i\} \in [0,1]$ represents the probability, that the value of the consequence $C_i$ caused by scenario $s_i$ is exactly $k$. In case where $C_i$ is continuously distributed, $\overline{C_i}$ is computed by

$$
\overline{C_i} = \int_{-\infty}^{\infty} C_i \cdot h(C_i) \cdot dC_i,
$$

(2.3)

where $h(C_i)$ is the probability density function of $C_i$ with

$$
Pr\{a \leq C_i \leq b|s_i\} = \int_{a}^{b} h(C_i) \cdot dC_i.
$$

(2.4)

Within this thesis, risk $\mathcal{R}$ is seen as the expected value (in the sense of statistics) of loss per time unit, where the unit of loss is ideally expressed in a monetary way. The unit of risk is then, e.g., [money/year] $\rightarrow$ ALE (annual loss expectancy).

If a scenario $s_i$ occurs, it is not sure that this will cause a consequence $C_i$, i.e., $p_i = Pr\{C_i|s_i\} < 1$. Therefore, the "consequence-frequency" $f_{c,i} = p_i \cdot f_{s,i}$ is used for risk computation. The risk $\mathcal{R}_i$ per scenario $s_i$ (see Equation 2.5) is defined as the product of the "scenario-frequency" $f_{s,i} \in \mathbb{R}_0^+$ (see Equation 2.7), the conditional probability $p_i$ and the mean consequence $\overline{C_i} \in \mathbb{R}$.

![Figure 2.1: Scenario frequency $f_{s,i}$, consequence probability $p_i$ and consequence frequency $f_{c,i}$](image)
2.1. The Concept of Risk

\[ \mathbb{R}_i = p_i \cdot f_{s,i} \cdot C_i \]  

(2.5)

\[ \mathbb{R} = \sum_{i=1}^{N} p_i \cdot f_{s,i} \cdot C_i, \]  

(2.6)

where \( i = \{1 \cdots N\} \in \mathbb{N} \) is the index of the scenarios \( s_i \) of the set \( S = \{s_1 \cdots s_N\} \) of all \( (N) \) scenarios, and \( C_i \) is the corresponding (monetary) mean consequence caused by the scenario \( s_i \). The "scenario-frequency" \( f_{s,i} \) of a given scenario can be estimated by \( f_{s,i} = \frac{n_i}{\Delta t} \), where \( n_i \) is the number of scenarios \( i \) which occurred during the time-period \( \Delta t \). If \( f_{s,i} \) is constant over time, it can be determined by the following limit:

\[ f_{s,i} = \lim_{\Delta t \to \infty} \frac{n_i}{\Delta t} \]  

(2.7)

Regarding the term risk, it could rise the question why it is necessary to break down the frequency into a "scenario-frequency" and a probability of consequence. To answer this question, it is appropriate to make an example: In today's home computer world, there dominate mainly two operating systems. With more than 90% market share, the Windows platform from Microsoft is almost a monopolist and the second operating system from Apple holds only about 6.6% of the market (see Table 2.1). Among the Apple users it is rumored that the Mac OS is much more resistant to computer viruses and worms than the Windows system. In fact, most Apple users do not have an antivirus software installed on their computers, but even so they never meet problems with malicious software.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Market Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>91.4%</td>
</tr>
<tr>
<td>Mac OS</td>
<td>6.6%</td>
</tr>
<tr>
<td>Linux</td>
<td>0.8%</td>
</tr>
</tbody>
</table>

Table 2.1: Market share of the home computer operating systems in September 2007 (Ref: marketshare.hitslink.com)

However, in 2006 the first Mac OS X virus called Leap-A demonstrated, that every system is vulnerable to such threats. A recent study [37] even demonstrated, that Apples operating system is more vulnerable to cyber-threats than the Windows platform, because the patching process of Microsoft is much better. Most probably, the reason for
the lack of more malicious software affecting Apple computers is not only because these system is much more resistant than the Windows platform, but it is also due to the fact, that Apples market share is negligible compared to those of the Windows platform. If a virus programmer can choose to write a malicious code which can potentially be spread epidemically on almost 90% of all computers worldwide, or alternatively to develop a code which probably affects only a few percent of the market, he normally chooses the first option. Further more, there exists much more people with extended skills in the handling of the Windows system, than Mac OS experts. As a result, the "scenario-frequency" $f_{s,i}$ of new malware for the Windows system is much higher than the one for the Mac OS system. In contrast, resistance of the two systems represented by the probability $p_i$ is possibly of the same magnitude. This applies to the mean consequence $\overline{C_i}$ as well. From this it follows that both systems are equivalent vulnerable to malware, but the Windows system is much more exposed to these kind of threats. To account this difference, the vulnerability and the exposure of a given system are introduced.

**Vulnerability** is the susceptibility of a system to a failure caused by a scenario $s_i$. Thus it is as a system's virtual openness to lose its design functions. In this respect, vulnerability could be defined as conditional probability $p_i$. Where a formal definition of risk is established among scientists, the idea of a quantitative vulnerability analysis is relatively new. However, e.g., a generic model for a quantitative vulnerability assessment by specifying vulnerability by the stability of a given system has been proposed in the past [38]. Within this thesis, the following definition of vulnerability is proposed:

$$V_i = p_i \cdot \overline{C_i}$$  \hspace{1cm} (2.8)

$$V = \sum_{i=1}^{N} p_i \cdot \overline{C_i}$$  \hspace{1cm} (2.9)

where $V_i$ is the vulnerability regarding $s_i$ and $V$ is the overall vulnerability of a system. To exemplify the difference between risk and vulnerability, let us assume that the "system" of interest is a person. The number of possible scenarios ($i$) that can harm seriously this person is large, e.g., he could be hit by a car on the street, he could be infected by a lethal disease, or he could be struck by, lightning. Regarding these events, every person is very vulnerable, but as the "scenario-frequency" $f_{s,i}$ of such events is very low, e.g., the risk of being struck by lightning is very low.
Exposure is a measure for the exposedness of a system and is defined within this thesis as the product of "scenario-frequency" and mean consequence.

\[ E_i = f_{s,i} \cdot C_i \]  \hspace{1cm} (2.10)

### 2.2 Classical System Analysis Methods

In this section, some of the established techniques of system analysis for risk evaluation are discussed. The aim of this section is to give a brief methodological overview and to clarify the causal connection of some technical terms. For a detailed description of the presented methods it is referred to literature, e.g., [5, 6, 39–41]. Up to now, the classical procedure of risk analysis (Fig. 2.2) starts with the identification of scenarios, followed by the estimation of their frequencies and consequences.

\[
\begin{pmatrix}
  s_1 \\
  s_2 \\
  \vdots \\
  s_{N-1} \\
  s_N
\end{pmatrix} \quad \Rightarrow \quad 
\begin{pmatrix}
  f_{s,1} \\
  f_{s,2} \\
  \vdots \\
  f_{s,N-1} \\
  f_{s,N}
\end{pmatrix} \quad \begin{pmatrix}
  p_1 \\
  p_2 \\
  \vdots \\
  p_{N-1} \\
  p_N
\end{pmatrix} \quad \begin{pmatrix}
  C_1 \\
  C_2 \\
  \vdots \\
  C_{N-1} \\
  C_N
\end{pmatrix}
\]

**Figure 2.2: Classical risk analysis procedure**

The established methods can be divided into two groups: Logic tree and tabular based models. Where the logic tree approaches only include the scenario rating, tabular based models often provide a complete risk analysis (and management) toolkit\(^1\).

#### 2.2.1 Logic Tree Analysis

In risk and reliability analysis, fault trees (FTs) and event trees (ETs) are well-known and widely applied type of Boolean logical trees. They are used for both qualitative and quantitative analysis and are often claimed by authorities. In addition FTs and ETs can be combined to analyze the scenario-likelihood parameters \(p_i\) and \(f_{s,i}\) (FT) and failure consequences \(\overline{C}_i\) (ET).

\(^{1}\)see for instance the Apis toolkits, www.apis.de
**Fault Tree Analysis**  A FT is the graphical representation of a Boolean equation. The idea of a FT is to determine all possible combinations (cut-sets) of lower-level failures leading to a specified system failure (top-event $s_t$) and to provide respective likelihood numbers ($p_i$ and $f_{s_i}$). Although there is a general lack of reliable data, the failure likelihood of individual system components is often assessed by best practice standards [9]. The graphical representation of this Boolean functions (FT) shows a tree structure of Boolean gates. The gates are Boolean operators (AND, OR) and they connect input events with the occurrence of a "higher"-level fault event. At the bottom of the tree, there are located the *basic events*, which are, e.g., hardware failures, human errors or computer software faults. Details about the FT methodology can be found in literature [6, 42].

In practice, FTs are applicable to very large systems (e.g., nuclear power plants) where the computation of the analytical solution of the top-event frequency is a NP-hard problem and therefore is derived by approximative algorithms, i.e., the *rare event approximation* [6] is applied. Another and more recent quantitative solution method uses transformations to BDD (binary decision diagrams), which are currently said to be the most efficient solution technique for static FTs. Nusbaumer 2007 [43] succeeded to compute the analytical solution of the core damage frequency of a nuclear power plant in Switzerland using BDDs.

**Event Tree Analysis**  In order to analyze and compute the mean consequence $C_i$ of a scenario $s_i$, the ETA (ET analysis) technique can be performed. An ET is a representation of a number of scenarios, starting from an *initiating event* (equivalent to scenario $s_t$) and leading through a chain of binary decisions (success/failure, only OR gates) to different *end-events* with different consequences $C_i$ [40]. ETA is a simple decision tree. An *initiating event* is postulated and it is tried to determine the corresponding effect on the overall system. The question of what system states are possible, starting from some initial condition, is important. Quantitative ETA demands for *conditional probabilities* for each successive binary choice in the chain to calculate the probability of the final system state. Feedback behavior or complex combinations of multiple failures are impossible to represent by ETs and they can become difficult to survey or check for freedom of error.
2.2. Classical System Analysis Methods

2.2.2 Tabular-based Methods

FMEA (Failure Mode and Effects Analysis) and HAZOP (Hazard and Operability Analysis) are two important non-mathematical, inductive and semi-formal methods where artifacts in the form of tables are created, documenting the results of the analysis process.

**FMEA** is a tabular based method where component *failures* are investigated inductively with respect to *effects* on sub-systems and the system as a whole. For this purpose a group of experts is usually decomposing a system into items, for which *failure modes* are determined. For each failure mode a qualitative judgment about its frequency is made. Further, thought is given about how such a failures can be *detected*, what *countermeasures* are available and what *consequences* are to be expected on neighboring components and on system level. The findings of the analysis are entered in tables and rated by a Risk Priority Number (RPN) which derived from the estimated frequency and consequence numbers. Several norms exist for FMEA forms in different fields of application [39, 44].

FMEA is mainly a qualitative method, when classifications are mapped to numerical values, it is called semi-quantitative. The advantage of FMEA is its relatively simple procedure, making it well suited for practical applications. It gives a good survey of installed components in a system (engineered system, plant or an operation), their failures and effects, and is eligible for investigation of rather complex systems. However, it is difficult to account for dependent, common cause and combinations of failures and proper analysis demands for rather large and costly resource allocation. In addition, the analysis has major subjective traits (expert judgment) and it is not assured, that all relevant scenarios are actually found.

**HAZOP** is a qualitative analysis method originated in the British chemical industry of the 1960s decade [41]. In essence it is an adaption of FMEA for process industry and geared toward operational aspects, but with applicability in other domains as well. With HAZOP, each item of a considered system is investigated toward possible deviations from normal behavior which could be problematic for the function of the system. Standardized guide-words are used to denote such process deviations. Again, results are summarized in tables, but in contrast to the FMEA method, HAZOP does not use a risk priority number to rate potential hazards. The goal of a HAZOP is hazard identification and determination of causes of production interruptions affecting plant productivity.
vantages and disadvantages are quite similar to FMEA as described above. Both FMEA and HAZOP have trouble when confronted with multiple failures, complex cause-effect relationships with nonlinear, coupled and feedback behavior and dynamic aspects in general.

2.3 Modeling and Simulation

The methods and techniques presented in this thesis, e.g., agent-based modeling (ABM) or business process modeling (BPM), all rank among modern and advanced scientific modeling approaches. There is a plurality of techniques in this field, developed by people with various scientific background. For this purpose, this section gives an overview of the most popular techniques and clarifies the corresponding terminology.

**Modeling:** The term *modeling* is often used across-the-board to denote every activity which is performed to "analyze" a system by means of a computer. In fact, this term lacks of an unique meaning, not to mention an unique definition. Within this thesis, the term stands for scientific modeling and is defined as follows:

**Def. 3** *Scientific modeling is the process of generating conceptual models,*

where a *conceptual model* is a theoretical construct that represents a system, with a set of variables and a set of logical and quantitative relationships (e.g., mathematical functions) between them. Conceptual models are constructed to analyze a system by an idealized logical framework. Idealized means that the model may make explicit assumptions that are known to be false (or incomplete) in some detail. Scientific models are characterized in different ways, e.g., by the *modeling perspective* or by the temporal *modeling approach*.

There are mainly two modeling perspectives: On the one hand, there is the *functional* modeling perspective, which emphasizes the use of functions that could stand on their own i.e., it illustrates the transformation from input to output. On the other hand the *object-oriented* modeling perspective focuses on the components (objects) of a system. The behavior of the system results from the collaboration of those objects. In literature, further modeling perspectives can be found, e.g., the structural, behavioral or rule perspectives.
The temporal aspect of a modeling approaches can also be divided in two groups: *Discrete Event Models* (section 2.3.2) and *Continuous Time Models* (section 2.3.3). The combination of this two approaches is often referred to as hybrid modeling.

![Diagram](image)

**Figure 2.3:** *Two classification types of scientific models*

**Visualization:** In order to simplify the process of modeling and to standardize the notation, several modeling "languages" have been developed, e.g., the object-oriented Unified Modeling Language (UML) [45] or the business process modeling notation (BPMN) [46].

**Model solving:** The basic goal of modeling is to improve the understanding of a given system and in particular to understand the system behavior over time. Regarding risk and reliability analysis, the output of a model should include information about frequencies and consequences of undesired scenarios. For this purpose, a formal model has to be designed and then solved by means of mathematical methods. *Continuous Time Models* are usually described by variables and equations (e.g., differential equations) and can be solved analytically or by numerical analysis (e.g., by applying a finite element or finite difference method). In contrast, *Discrete Event Models* are formulated by
means of logic algorithms and often they cannot be solved analytically or by numerical methods. Some statistical methods help to make quantitative statements, but in general discrete event models are analyzed by means of computer simulations.

**Def. 4 Simulation:** A computer program that attempts to imitate the behavior of a scientific model (see Def. 3) of a particular system.

Thus, a computer simulation is a virtual experiment of a real situation. In recent years, they have become an important part of scientific modeling to gain insight into the operation of systems, or to observe their behavior without performing physical tests. Thereby, the generation of random numbers plays an important role. In some disciplines\(^2\), modeling approaches are classified due to the fact, that they use random number generation to solve the model, i.e. it is distinguished between probabilistic models and deterministic ones. To include this point of view, Random Number Models are discussed below.

### 2.3.1 Random Number Models

Many computer simulations need random input, which produce output that is itself random. Therefore, random numbers and random variables must be generated for the simulation experiment. The quality of these random numbers is an important issue. A computer is a deterministic machine, only carrying out instructions feed beforehand in the form of a program. All computer generated random numbers are therefore produced according to some deterministic algorithm, which should have the property of producing numbers that resemble true random numbers to a great extent. In fact, the numbers are only "pseudorandom" numbers (in the following, they are nevertheless called "random" numbers). Mathematicians have developed very good algorithms for random number generation that are well established and tested (e.g., \([47]\) and \([48]\)).

Although the generation of random numbers generally is part of the *model solving approach* (simulation), in some disciplines random numbers denote a separate class of models. The classification of these random number models is difficult as different disciplines use different terms. In the field of reliability engineering, random number models are mostly called *Monte Carlo Models*, but also *Markov Models* make use of random number techniques when they are solved by simulation. In financial and insurance mathematics, the term *Stochastic Model* is used for simulations based on random num-

---

\(^2\)In reliability analysis, models using random numbers are called Monte Carlo models, where in the field of operations research, they are rather referred to as stochastic models.
ber generation. In fact, there are many more simulation techniques making use of random number generation including ABM, some Discrete Event Models and in addition, random numbers are even used as stochastic input for Continuous Time Models.

![Classification of modeling approaches](image)

**Figure 2.4:** Classification of modeling approaches

### 2.3.2 Discrete Event Models

In discrete event model, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system. For example, if an elevator is simulated, an event could be "level 6 button pressed", with the resulting system state of "lift moving" and eventually (unless one chooses to simulate the failure of the lift) "lift at level 6".

A number of mechanisms have been proposed for carrying out discrete event simulation, among them are the event-based, activity-based, process-based and three-phase approaches [49]. The three-phase approach is used by a number of commercial simulation software packages, but from the user's point of view, the specifics of the underlying simulation method are generally hidden. For a detailed introduction to discrete event systems, see [50].
2.3.3 Continuous Time Models

In contrast to the *Discrete Event Models* where state variables change at discrete times in time, a continuous-time model is one in which the state variables change continuously over time. It is represented through a set of state equations (often differential equations) with initial conditions specified. The state equations imply the behavior of all the system state variables over time. If the equations are not analytically solvable, they are treated with numerical methods (e.g., discretization and numerical integration, see [51]).

2.3.4 Risk Number Computation

In contrast to the screening methods described in section 2.2, simulation models are not a priori designed for risk analysis. As computer simulations are equivalent to virtual experiments, the desired variables (e.g., frequency and consequence) have to be defined and then observed during a simulation experiment. In general, there are two groups of simulation experiments: (a) "*infinite* time simulation" and (b.) *finite time simulation with variable initial conditions*. In case (a.), it is assumed that the item of interest will stand forever as long as recovery processes are considered.

![Figure 2.5: Example of a system variable (performance) as an observable for consequence analysis](image)

During the simulation, key-variables (e.g., system performance, see example in Fig. 2.5) are observed and if they reach a certain threshold (e.g., performance bellow 95%), the simulation algorithm starts to record the consequence $C_i$ of this low performance (*consequence quantification*) for the whole duration until the system reaches its normal state due to recovery processes. In parallel, the cause (scenario $s_i$) for this consequence is identified (*consequence identification*).
The longer this kind of simulation experiment is performed, the more undesired scenarios are observed and the better is the statistics of the measured consequence frequencies $f_{c,i} = p_i \cdot f_{s,i}$ and consequences $C_i$.

In case (b), it is assumed that the item of interest will collapse at some time in the future, i.e., recovery processes are mostly neglected. In contrast to case (a), the simulation experiment is stopped when the collapse occurs. Then, the time to collapse (time to failure) $TTF$ and the emerged consequence $C_i$ are recorded before the simulation experiment is restarted for a next run. The more simulation runs are performed, the better is the statistics of the measured times to failure $TTF$ and consequences $C_i$. The consequence frequency $p_i \cdot f_{s,i}$ can then be derived from $TTF$. 

Figure 2.6: Classical risk analysis procedure

Figure 2.7: The concept of deriving frequencies and consequences by "infinite" time simulation. The undesired events emerge during the simulation experiment.
2.4 Information Systems and Risk Analysis

2.4.1 Information System

The term information is used with differing meanings and is commonly used attributively to refer to intelligence or news concerning some particular fact, subject or event (see Oxford English Dictionary). Generally speaking, the concept of information is closely related to notions of data, knowledge, or wisdom [52, 53].

**Def. 5** *Data is the raw material of information as for instance in the form of facts and figures obtained from experiments or surveys, that itself does not have a meaning.*

**Def. 6** *Information is processed data, which has been given a meaning by means of rational connection as for instance collected facts about a specific subject. Provides answers to 'who', 'what', 'where', and 'when' questions.*

**Def. 7** *An information system (IS) is a system that processes information, which includes the recording, transmitting, transforming, storing, and retrieval of information.*

As the *processing* of data is the key attribute of an IS, the specification of the corresponding processes is essential when trying to model and analyze a given IS.

2.4.2 State of the Art in IS Risk Analysis

The problem of computer security arised in the mid-60’s when people started to share computer systems [54]. A historical overview of the emerged methodologies and standards to secure IS is given, e.g., by Diergardt [25]:

"The desire to manage computer security issues with a risk based approach arose in the early 1970s. In the same period risk based approaches also emerged in other fields, e.g., nuclear safety. According to [55] these early approaches focused particularly on risk analysis of natural disasters, physical theft and sabotage. The principle document in this area is the Guideline for Automatic Data Processing Physical Security and Risk Management" (FIPS PUB 31) [56]. It quantifies risk with an annual loss expectancy, which is measured in Dollars, and gives some guidelines for acquiring the appropriate values and probabilities."

As the FIPS PUB 31 provides little guidance in regards to risks emerging from data and programs in computer systems the National Bureau of Standards (NBS) "engaged
2.4. Information Systems and Risk Analysis

Courtney and Reed to develop the Guideline for Automatic Data Processing Risk Analysis (FIPS PUB 65) [21]. The core of the FIPS PUB 65 formed the *Annual Loss Exposure* (ALE). The ALE is the product of the estimated impact $I$ in dollars and estimated frequency of occurrence $F$ per year. Since it is hard to determine exact numbers, the estimates for impact and frequency had to be specified in orders of magnitude. OMB A-71 and FIPS PUB 65 are considered to be the initial incentive for applying risk analysis techniques to computer systems in government and private sectors." [25]

Later on a trend to automated information risk analysis is noticed. Among others, Bayesian Decision Support Systems (BDSS) [57], Lawrence Livermore Laboratory Risk Analysis Methodology (LRAM) [58, 59], Risk Analysis and Management Methodology (CRAMM) [18, 19], Los Alamos Vulnerability and Risk Assessment Methodology (LAVA) [60] and the *Methodologie d’Analyse des Risques Informatiques et d’Optimisation par Niveau* (MARION) [20] have been developed. "Simultaneously the National Institute of Standards and Technology (NIST) seeded research and organized the International Computer Security Risk Management Model Builders Workshop, e.g., [60], in order to establish a general risk management framework for IS. One achievement of the workshops was the introduction of a conceptual model which identified constituent functional elements for IS risk management, as for instance *uncertainty analysis*. The Guide to Security Risk Management [61] and the Guidelines for the Management for IT Security (GMITS) [62] and related documents can also be considered as indirect products of these workshops. However, the workshops did not succeeded in establishing a framework which was able to provide a method for measuring risk management techniques and to ensure their comprehensiveness." [25] A trend to a divergence from risk analysis approaches applied in practice to the ones developed in the academic environment is observed in the 1990s which still holds today. Where the adaptation of object oriented approaches for risk analysis was the focus in researchers, as, e.g., [63, 64] or larger projects as CORAS [22] or at the Sandia National Laboratories [65], National IT security agencies and standard organizations started to publish their own risk management frameworks, as, e.g., the German Information Security Agency (GISA) [66], the Communication Security Establishment [61], the Standards Australia and the Standards New Zealand [67], International Standard Organization [62, 68–71], NIST [72], among others.

Despite all these efforts, the application of codes of practice is established in practice. Although these guidelines are restricted in regards to the IS they cover and do only answer limited set of questions in regards to risk analysis which a decision maker may have. Popular codes of practice are the IT-Baseline Manual [66], the Code of Practice for Information Security Management [3] and the Information Technology Infrastructure
Library (ITIL).

ITIL for example is a TSO\(^3\)-directive used by many thousands of organizations around the world, which represents a set of concepts and techniques (including risk and availability analysis) for managing IT infrastructure, development, and operations. The TSO claims that ITIL is the only consistent and comprehensive documentation of best practice for IT Service Management. For deciding on the appropriate resource allocation for safeguards and security concepts for IS, the concept of risk, in particular quantitative techniques, are increasingly accepted [73]. Nevertheless, the state of the art of risk analysis techniques is not satisfactory and leads to the fact that in practice the codes of practice predominate.

\(^3\)TSO, or The Stationary Office, are the actual publishers of the current ITIL volumes. They publish in excess of 15,000 titles per year.
Chapter 3

Business Process Modeling for Risk Analysis

3.1 Business Processes

An information system (IS) is characterized as a system that processes information which includes the recording, transmitting, transforming, storing, and retrieval of information [25], i.e., an IS is not necessarily restricted on computer network implementations. The enterprises’ perspective is given by business processes (BP). A BP is defined as “a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships" [74].

3.1.1 BP Modeling Techniques

Thus, an IS is represented by a business process model (BPM). There are many BPM techniques in use, e.g., Flow Charts, Block Diagramms or Petri nets. In the literature, the reader will find lists of techniques and tools whereas these approaches are often compared with each other under different perspectives (e.g., generic methodological structures [75], [76]; IS security analysis and design [77], software tools supporting BPM [78]).
3.1.2 Low Acceptance of Petri Nets

Among scientists, Petri Nets (PN) are widely used for modeling and simulation of complex systems since its introduction in 1962 [123], [124]. PN are highlighted in this context, as they are considered as a promising tool in risk and reliability analysis, and PN-modelling has reached a sophisticated methodological level. However, it is the authors' experience with major (Swiss) enterprises that PN do not reach the level of accepted practicability in non-academia application areas. This impression is fleshed out by a literature research although private enterprises are usually less interested in publishing than university-level institutions. The spot check is sampled from [95], where the "EI COMPENDEX database covers the core literature of the engineering field, including related specialties and technologies. Citations are drawn from approximately 400 journals; key conference proceedings and abstracts are also included". The Boolean search criterion [(petri net* AND (reliability OR availability) AND analysis) AND (PY >1999)] results in 96 hits back to the publication year 2000. According to the first authors’ affiliations, these papers stem from universities (70 papers; 72.9%), research centers (8; 8.3%), private enterprises (10; 10.4%), universities of applied sciences, schools, etc. (4; 4.2%), and others (4; 4.2%). In summary, about 85% trace back to academia papers.

The 70 university papers stem from 48 universities, whereby 12 universities provide at least two papers (altogether 34 papers), i.e., about half of the academia papers and about 35% of all papers. Major distributors are the University of Illinois at Chicago (6 papers) and the National Chin Yi Institute of Technology and Taichung (Taiwan) (5 papers). Only six of the papers' abstracts refer to the keyword "industry" whereby half of them deal with information and communication technology (ICT). In summary, there are only a few mainspring universities developing, supporting and implementing the PN methodology; practitioners are reserved.

In dependency of the stochastic nature of reliability analysis, the 96 papers only mention a few types of PN (multiple entries), namely stochastic PN (31 hits), timed (6), colored (6), high level in general (5), and other (8). The papers’ abstracts rarely name (software) tools. SPNP (Duke University) is mentioned four times. Other tools are, e.g., GreatSPN (University of Torino); SHARPE (Duke University); Artifex (RSoft Design Group), and SMART (University of California). As a result of our Lab's research activity, GreatSPN is also recommended as a PN package for risk and reliability analysis of computer networks [125].

The stochastic PN usage (or formal methodologies in general) is mainly restricted to academia where PN are successful [126], [127]. The combinatorial growth of the state
space of stochastic PN is a major restraint of this formalism when analysing real systems [128]. In particular, industry’s acceptance of stochastic PN may be reduced by the various extensions of PN tools leading to very different approaches, modelling and simulation qualities as well as analysis properties. Finally the lack of (stochastic) PN standards is considered as deterring to practitioners [126]. Practical PN applications to risk and reliability analysis are given in [129], [130], [131], [132]) among others. Constraints in applied ICT risk analysis gives [12], e.g., a three to maximum six month time limit to execute a (customary) risk analysis.

In summary, some PN’s problems of system modelling for risk analysis purposes are identified as

- methodological ponderosity in modelling complex systems (e.g., the simultaneous consideration of time, "coloured tokens", stochastic properties, and consequences)

- intricateness in computing risk figures

- low acceptance of PN in practice.

In order to overcome these problems, object-oriented techniques are considered as promising.

### 3.1.3 Specified Requirements

In order to identify an appropriate technique for risk analysis purposes, common techniques can be classified and rated. Which BPM technique comes into operation within this thesis is mainly influenced by a set of selection criteria or requirements [79]:

- **Process View**: To facilitate the modeling business practitioners, the inclusion of a process view (for business process modeling) is essential.

- **Object View**: To enable the future integration in a agent-based model, the inclusion of an object view is required.

- **Correctness**: The modeling technique has to be complete and consistent.

- **Standardization**: The technique should enable a standardized translation to simulation code.
• **Acceptance**: The acceptance of modeling language among business managers and IT specialists is important. Uncommon techniques are rejected by the business practitioners.

• **Simplicity**: A comprehensible and clear modeling technique is desired.

• **Tool Support**: There are many tools supporting the design and analysis of BPM. A technique without tool-support is unacceptable today.

• **Availability**: Proprietary technologies should be avoided.

Using the eight criteria above, seven of the most established BPM techniques have been rated based on a qualitative evaluation. The result of this rating is presented in Table 3.1, where 1 represents a positive, 0 a neutral and -1 a negative valuation.

<table>
<thead>
<tr>
<th></th>
<th>BPMN</th>
<th>PN</th>
<th>EPC</th>
<th>ARIS</th>
<th>ERM</th>
<th>UML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process View</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Object View</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Correctness</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Standardization</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Acceptance</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Simplicity</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tool Support</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Availability</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

None of the techniques in Table 3.1 use risk related approaches. The rating bases on a qualitative evaluation and it helps to identify a suitable candidate to be extended for risk analysis purposes. As a result of this rating, PN (Petri-nets) and ERM (Entity Relationship Modeling) are low rated. Where Petri-nets are remaining a scientific tool with less acceptance among practitioners [80], ERM lack of availability, tool-support and standardization. FC (Flow Charts), EPC (Event-driven Process Chains) and ARIS (Architektur integrierer Informationssysteme) are widely used (e.g., ARIS and EPC often precede the implementation of SAP/R3) and mostly consistent, but they have other shortcomings, e.g., high costs for software licenses. Finally, The Unified Modeling Language (UML) and the Business Process Modeling Notation (BPMN) remain as the best rated techniques. UML is particularly designed for the modeling of software architecture.
and includes different types of diagrams, which makes its application more challenging. In contrast, the BPMN is an advanced BPM standard which is designed for modeling IS.

3.2 Introduction to BPMN

This section provides a basic introduction to BPMN selected as modeling technique within this thesis. The notation is described in order to be able to develop an extension of the BPMN for risk analysis.

The BPMN is a standard developed by a task force of the Object Management Group (OMG\(^1\)), the so-called Business Process Management Initiative (BPMI\(^2\)). The intention was to provide an universally understandable notation. Similar to the UML, which was also developed by an OMG task-force, it is designed as a bridge from the business analysts and developers that designs the BP, to the software engineer responsible for the technical implementation. The design of BPMN is realized in such a way, that executables can be generated automatically. The Business Process Diagram (BPD) of BPMN is based on the Flow Chart technique, and tailored to the modeling of IS.

3.2.1 Basic BPMN Elements

The BPD of BPMN consists of a set of graphical elements. With this elements, a modeler can develop simple diagrams that look familiar to business analysts. Activities are formed as rectangles and decisions have a diamond shape. The graphical elements of the notation are organized in specific categories. This provides that the reader of a BPD can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram. The four basic categories of elements are Flow Objects, Connecting Objects, Swimlanes and Artifacts. According to [140], these categories have the following structure:

**Flow Objects:** The BPD of the BPMN has three core elements, the so-called *Flow Objects*. These are events, activities and gateways:

---

\(^1\)www.omg.org  
\(^2\)www.bpmi.org
Events are represented by circles. They stand for things that "happen" during the execution of a BP. Events in general have a cause or a consequence and thus they affect the flow of a BP. The default event is an empty circle. In order to differentiate between several kinds of events, the circles are extended with internal markers. With regard to the BP flow, events are grouped into three categories: Start events, intermediate events and end events.

To represent activities, rounded-corner rectangles are used. An activity is thought to represent work or task which has to be performed within a BP. There are two types of activities: The Task and the sub-process, where the sub-process is a single activity symbol (including a plus sign) which summarizes a whole sub-process.

Diamond shapes are used to represent gateways. Their role is to control the sequence flow within a BP by splitting or merging it. Gateways usually represent decisions or conditional flow-branches. Markers are used to specify the type of gateway, e.g., an X is used for an x-or gateway.

**Connecting Objects:** By connecting the *Flow Objects* together into a single diagram they represent a basic graph structure of a BP. Mainly three kind of *Connecting Objects* are known in the BPMN formalism: Sequence Flow, Message Flow and Associations.

In order to indicate the order (sequence), in which activities of a BP are performed, the flow objects are connected by solid lines comprising a solid arrowed head. This represents the sequence flow. In contrast to other modeling languages, the term "control flow" is not used.
3.2. Introduction to BPMN

Message Flow  The message exchange within a BP and between process participants is represented by open arrowed, dashed lines. This arrows are called *message flow*. To distinguish between different participants, separate pools are used in the BPMN diagrams (see Swimlanes).

Association  The association between flow objects and data, text or other artifacts is illustrated by association arrows (dotted line, line arrow). The associated artifacts specify the input and output of the flow objects.

Swimlanes  In order to organize a BP and to illustrate responsibilities of different participants, the use of swimlanes is popular in several process modeling languages. Swimlanes can be used to represent the separate process of each participating agent which may be persons, organizations or even hardware components. Thus, swimlanes add an object oriented view to the BPMN model. Two kinds of swimlanes are used:

- **Pool**  Process participants are represented by grouping their activities in separate graphical areas, so called pools. Pools are also used to separate associated processes, e.g., a B2B process within an organization.

- **Lane**  To further specify and allocate process ownership or to categorize and organize activities, pools can be divided into subpartitions, so called lanes. Lanes divide a pool by a horizontal or vertical line.

Artifacts  The BPMN is a modern and very flexible modeling notation. It allows a modeler to extend the basic notation and adapt it to specific situations. To enrich a BPM with numerous additional information, different kinds of artifacts can be added to the model diagram. The use of artifacts helps to make a BPD more readable and complete. In the BPMN formalism, three types of artifacts are specified:
Data Object

Whenever data is produced or required by a BP activity, this is illustrated by associating a data object to the activity (connected by the association arrow).

Group

For in detail analysis and documentation purposes, the BPMN objects can be further grouped by surrounding them by a rounded corner rectangle (dashed line). This kind of grouping has no influence on the process flow.

Annotation

Notes, remarks, additional information can be added to any type of BPMN object by making an annotation. Annotations have no impact on the flow of a process.

3.3 Extending BPMN for Risk Analysis

In order to analyze risk by use of BPMN, it is essential to extend the corresponding modeling technique. As BPMN’s are success oriented, an adapted BPMN risk analysis method has to comprise undesired events and to map their frequencies and consequences.

![BPMN example](image)

**Figure 3.1: BPMN example**

A first approach is to extend the activity element of the BPMN formalism (Figure 3.1) with the option to fail. The failure of an activity leads to a *Failure Event* \( F_1 \). The potential undesired outcome of such a failure is determined by an *Event Tree* (Figure 3.2). The
probability $P_{F,i}$ of a malfunction is estimated by applying a fault tree analysis (FTA) and thus analysis the influence of the IT-System which supports the corresponding BP. A similar approach was proposed by [81] in the field of logistics.

This detailed analysis is only done for (prioritised) activities $A_i$, $(i = 1, \ldots, n)$. As a BP can be carried out in various paths, the BPMN of a BP consists of $k = 1, \ldots, m$ paths $M_k$. According to Figure 3.2, each failure $F_i$ of an activity $A_i$ can result with the probability $P_{F,i} = Pr(C = C_{F,i,j}|F_i)$, in $j = 1, \ldots, v$ different consequences $C_{F,i,j}$. The expected value $C_{F,i}$ of the consequence is calculated according to Eq. 3.1. In order to work out a meaningful risk value for each activity, it is essential to include the average rate of activity execution $f_{A,i}$ [year-1]. The more frequent an activity is processed, the higher is its error rate, i.e., $f_{A,i} \cdot P_{F,i}$ and its risk (Eq. 3.2), where $P_{F,i}$ is the probability of failure $F_i$.

\[ C_{F,i} = \sum_{j=1}^{v} P_{F,i,j} \cdot C_{F,i,j} \quad (3.1) \]

\[ R_{F,i} = f_{A,i} \cdot P_{F,i} \cdot C_{F,i} \quad (3.2) \]
The risk value $R_{Fi}$ represents the expectation of the unplanned expenses per year caused by the failure $Fi$ of activity $Ai$. $C_{Fi}$ stands for the expected loss, if the activity $Ai$ fails once. $C_{Fi}$ could also be interpreted as a risk value, but in order to follow the risk definition according to Equation 3.2 for every level of abstraction, it is called a consequence. Considering a complex BP with feedback loops, the most challenging task is to determine $f_{Ai}$ for each activity. In order to overcome this difficulty, it is assumed that each path $\vec{M}_k = (M_{1,k}, M_{2,k}, \ldots, M_{n,k})$ is passed only once. $M_{i,k}$ denotes, how many times $Ai$ is processed following the path $\vec{M}_k$. Consequently $M_{i,k}$ is an integer where $M_{i,k} = 0$ denotes that activity $Ai$ does not occur in path $\vec{M}_k$, and $M_{i,k} = 1$ indicates that $Ai$ is passed once along path $\vec{M}_k$. Depending on the availability of processing statistics it might be easier to estimate the process rate of each path $f_{M,k}$ than the rate $f_{Ai}$ of each activity. In Equation 3.3, the relationship between $f_{M,k}$ and $f_{Ai}$ is given, where $\vec{f}_A = (f_{A,1}, \ldots, f_{A,n})$ and $\vec{f}_M = (f_{M,1}, \ldots, f_{M,m})$:

$$\vec{f}_A = M \cdot \vec{f}_M$$

(3.3)

$$f_{Ai} = \sum_{k=1}^{m} M_{i,k} \cdot f_{M,k}$$

(3.4)

The matrix $M(m \times n)$ consists of the path vectors $(\vec{M}_1, \ldots, \vec{M}_k)$ respectively the matrix elements $M_{i,k}$. To exemplify the above equations, a simple process (Fig. 3.3) containing one loop is considered:

![Figure 3.3: Sample process containing one loop.](image)

It is assumed approximatively, that the number of paths through the BP is limited. The simplest path in this example would be $A_1 \rightarrow A_2$, i.e. $\vec{M}_1 = (1, 1, 0)$. When passing the loop once, $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_2$ is the corresponding sequence of processed activities, i.e. $\vec{M}_2 = (1, 2, 1)$. 
3.3. Extending BPMN for Risk Analysis

\[
M = \begin{pmatrix}
1 & 1 \\
1 & 2 \\
0 & 1
\end{pmatrix}, \quad \vec{f}_M = \begin{pmatrix}
10 \\
5
\end{pmatrix}
\] (3.5)

Assuming that \(\vec{M}_1\) and \(\vec{M}_2\) are by far the most frequently occurring paths through the BP, all other paths are neglected. Assuming further more, that \(\vec{M}_1\) is processed on average 10 times a day and \(\vec{M}_2\) only 5 times, the process rate vector \(\vec{f}_M\) can be provided (Eq. 3.5). Knowing all paths \(\vec{M}_k\), which gives the Matrix \(M\), and their rates, it is possible to compute the process rates per activity \(\vec{f}_A\).

\[
\vec{f}_A = M \cdot \vec{f}_M = \begin{pmatrix}
15 \\
20 \\
5
\end{pmatrix}
\] (3.6)

Eq. 3.6 denotes, that on average \(A_1\) is processed 15 times, \(A_2\) 20 times and \(A_3\) 5 times per day. These rates \(f_{A_1}\) are needed to compute the risk per activity (Eq. 3.2). In order to obtain an overview of the different risks of a BP, the methodology is structured in 6 working steps.

**Step 1, BPM:** A BPMN model of a BP is generated.

**Step 2, Prioritization:** The BPMN model of step 1 consists of important and less important activities. Each \(A_i\) is weighted on an expert judgment procedure. The results are combined in a prioritization number \(S_i\).

\[
S_i = W_i \cdot J_i
\] (3.7)

The weighting factor \(W_i\) bases on a classification scheme that considers the level of control of the enterprises' accurate activity execution from "absolutely sure" (1) to "unsure" (5). The judgment factor \(J_i\) indicates the activity's importance for that enterprise and is classified in "extreme important" (5) to "unimportant" (1). As a result, each activity \(A_i\) is characterized by a prioritization number of minimum value 1 to maximum 25.

**Step 3, Paths:** As mentioned above it is assumed, that a BP may be successfully executed by a limited number of \(k\) different paths \(\vec{M}_k\). These paths have to be identified and the process rate \(f_{M,k}\) of each path has to be assessed (see example above). This
is done preferably by analyzing process statistics or alternatively by expert judgment.

**Step 4, Activities’ risks:** For prioritized activities, the FTA and ETA techniques are used in order to assess the failure probability and corresponding consequences $C_{F,i}$. With Eq. 3.4 the rates $f_{A,i}$ are computed to finally establish the risk $R_{F,i}$ of activity $A_i$ according Eq. 3.2.

**Step 5, Paths’ risks:** As mentioned above, a BPM may contain XOR operators or loops, i.e., the overall BP can follow $k$ flow paths from start to completion. The success probability $P_{M,k}$ of a path results from Eq. 3.8.

$$P_{M,k} = \prod_{i=1}^{n} (P_{F,i})^{M_{i,k}} \tag{3.8}$$

Where $P_{F,i} = 1 - P_{F,i}$ is the success probability of activity $A_i$ and the failure probability of the path $M_k$ is indicated as $P_{M,k} = 1 - P_{M,k}$. If a specified activity is passed because of a loop, the second transit is considered as an additional and independent activity. The summed up consequences along a path $k$ are computed by

$$C_{M,k} = \frac{\sum_{i=1}^{n} M_{i,k} \cdot P_{F,i} \cdot C_{F,i}}{\sum_{i=1}^{n} M_{i,k} \cdot P_{F,i}} \tag{3.9}$$

Equation 3.8 and 3.9 finally result in the path’s risk $R_{M,k} = f_{M,k} \cdot P_{M,k} \cdot C_{M,k}$ following Eq. 3.2.

**Step 6, BP’s risk:** Having calculated each success probability $P_{M,k}$, the total failure probability of a BP is given by

$$P_{BP} = 1 - \frac{\sum_{k=1}^{m} f_{M,k} \cdot P_{A,k}}{\sum_{k=1}^{m} f_{M,k}} \tag{3.10}$$

The corresponding consequences results from
Both risk parameters are related to the execution rates \( f_{M,k} \) or \( f_{A,i} \) as increasing runs boost the failure frequency. Finally, the total risk of a BP is estimated by \( R_{BP} = f_{BP} \cdot P_{BP} \cdot C_{BP} \) following Equation 3.2, where

\[
C_{BP} = \frac{\sum_{i=1}^{n} f_{A,i} \cdot P_{F,i} \cdot C_{F,i}}{\sum_{i=1}^{n} f_{A,i} \cdot P_{F,i}}
\]  
(3.11)

\[
f_{BP} = \sum_{k=1}^{m} f_{A,k}
\]  
(3.12)

3.4 Case Study

The approach was tested in and adopted for the business framework of a major Swiss Internet provider [82]. Business constraints enforce further simplifications concerning the quantification of model parameters. In Step 1 the BP "Software Security Vulnerability Management" (SSVM) is represented by a BPMN model (see Fig. 3.4). Within this process, the provider focuses on the detection and elimination of software vulnerabilities. SSVM is a supporting process which shows activities how to react on vulnerabilities in software systems after (potential) exploits. SSVM mainly consists of effective and efficient management of hot fixes and patches. SSVM is structured in five major procedural steps A to E with 15 corresponding activities \( A_i \):

- **A: Identification**, \( A_1 \): identify vulnerabilities
- **B: Analysis**, \( A_2 \): allocate vulnerabilities, \( A_3 \): suggest solutions, \( A_4 \): estimate risks, \( A_5 \): evaluate systems and applications.
- **C: Elimination**, \( A_6 \): make up a team, \( A_7 \): elimination of vulnerabilities.
- **D: Elimination control**, \( A_8 \): control vulnerability elimination, \( A_9 \): balance functionality vs. safety, \( A_{10} \): check out acceptability, and \( A_{11} \): document acceptability. Finally, recurring procedures are defined.
- **E: Ticket handling**, \( A_{12} \): generate, \( A_{13} \): route, \( A_{14} \): annotate information, \( A_{15} \): complete.
The ticket handling activities are considered at least as very reliable \( (\bar{P}_{F,i} \geq 0.99) \). For that reason, the cases study omits them in the risk estimation procedure.

![Diagram](image)

**Figure 3.4:** BPMN model of the "Software Security Vulnerability Management" process of a major Swiss Internet service provider. For the activities \( A_1 \cdots A_6 \), the process is linear and therefore summarized by a single activity.

The quantification of parameters comprises Steps 2 to 5. The results of expert judgments and computations are summarized in Table 3.3. The prioritization due to Step 2 gives five activities \( A_1, A_4, A_5, A_7, \) and \( A_9 \) (shadowed in Table 3.3) which are analyzed in detail by an ETA/FTA approach. The approach is exemplified by \( F_i \) (Fig. 3.5). For the other activities, \( P_{F,i} \) and \( C_{F,i} \) are estimated by expert judgment, where Table 3.2 was used for probability estimation.

<table>
<thead>
<tr>
<th>Activity ( i ) is operated</th>
<th>Success: ( \bar{P}_i )</th>
<th>( P_i = 1 - \bar{P}_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolutely reliable</td>
<td>0.9999</td>
<td>0.0001</td>
</tr>
<tr>
<td>very reliable</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>reliable</td>
<td>0.95</td>
<td>0.05</td>
</tr>
<tr>
<td>fairly reliable</td>
<td>0.68</td>
<td>0.32</td>
</tr>
<tr>
<td>unreliable</td>
<td>0.52</td>
<td>0.48</td>
</tr>
</tbody>
</table>

In order to compute the rates of activity execution \( \bar{f}_A \), the five most commonly executed process paths \( \bar{M}_1 \cdots \bar{M}_5 \) are identified by expert judgment. The process rates \( f_{M,k} \) of each path were estimated by combining statistical process data and expert judgment. By using Eq. 3.3, \( \bar{f}_A \) was computed as follows:
Figure 3.5: ETA of the failure event $F_i$ of Activity $A_i$. The resulting average consequence $C_{F_i} = 27$.

\[
\tilde{f}_A = M \cdot \tilde{f}_M = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 2 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix} \begin{pmatrix}
100 \\
10 \\
45 \\
20 \\
5 \\
70 \\
50 \\
45
\end{pmatrix} = \begin{pmatrix}
180 \\
180 \\
180 \\
180 \\
180 \\
215 \\
145 \\
70 \\
45
\end{pmatrix}
\] (3.13)

The results of activity and path analysis are summarized in Table 3.3. For the gray labeled activities, $P_{F_i}$ and $C_{F_i}$ are obtained with a FTA and an ETA respectively. Interestingly, the highest risk ranking is allocated at an activity ($A_2$: allocating vulnerabilities), which was not identified by the prioritization of the activities which was done in Step 2. This may have two reasons: On the one hand, the estimation of expectancy values ($C_{F_i}$) by experts is always problematic, especially if the distribution of the esti-
Table 3.3: Summary of the Case Study results

<table>
<thead>
<tr>
<th>$A_i$</th>
<th>$W_i$</th>
<th>$J_i$</th>
<th>$S_i$</th>
<th>$P_{F,i}$</th>
<th>$f_{A,i}$</th>
<th>$C_{F,i}$</th>
<th>$R_{F,i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>0.01</td>
<td>180</td>
<td>52</td>
<td>94</td>
</tr>
<tr>
<td>$A_2$</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>0.05</td>
<td>180</td>
<td>50</td>
<td>450</td>
</tr>
<tr>
<td>$A_3$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.0001</td>
<td>180</td>
<td>25</td>
<td>0.5</td>
</tr>
<tr>
<td>$A_4$</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>0.05</td>
<td>180</td>
<td>27</td>
<td>241</td>
</tr>
<tr>
<td>$A_5$</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>0.05</td>
<td>180</td>
<td>19</td>
<td>172</td>
</tr>
<tr>
<td>$A_6$</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>0.01</td>
<td>180</td>
<td>75</td>
<td>135</td>
</tr>
<tr>
<td>$A_7$</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>0.01</td>
<td>215</td>
<td>100</td>
<td>215</td>
</tr>
<tr>
<td>$A_8$</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0.01</td>
<td>145</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>$A_9$</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>0.05</td>
<td>70</td>
<td>56</td>
<td>196</td>
</tr>
<tr>
<td>$A_{10}$</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0.01</td>
<td>50</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>$A_{11}$</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0.01</td>
<td>45</td>
<td>50</td>
<td>23</td>
</tr>
</tbody>
</table>

Estimated value has a high deviation. On the other hand, the risk value is not exactly the same as the prioritization number $S_i$, but the two numbers have only a correlation (see Fig. 3.6). Besides the one of $A_2$, the risk ratings reflect more or less the expectations.

![Figure 3.6: Correlation between the prioritization number $S_i$ and the risk number $R_{A,i}$](image)

The coefficient of correlation is $r^2 = 0.6$ if the outlier of $A_2$ (black point) is neglected.

To analyze the five most common paths separately, Table 3.4 depicts the risk relevant numbers per path. This view helps to decide, which path should be prevented to reduce the overall risk.

In Step 6, the results of activity and path analysis are combined to compute the overall SSVM risk. Eq. 3.10 gives the SSVM failure probability $P_{BP} = 0.197$. The corresponding consequence parameter is $C_{BP} = 42.0$. The SSVM is either characterized by
Table 3.4: Summary of the Case Study results

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>$M_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{M,k}$</td>
<td>100</td>
<td>10</td>
<td>45</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>$P_{M,k}$</td>
<td>0.18</td>
<td>0.19</td>
<td>0.22</td>
<td>0.22</td>
<td>0.23</td>
</tr>
<tr>
<td>$C_{M,k}$</td>
<td>39.8</td>
<td>43.2</td>
<td>43.5</td>
<td>45.5</td>
<td>45.6</td>
</tr>
<tr>
<td>$R_{M,k}$</td>
<td>703</td>
<td>83</td>
<td>441</td>
<td>205</td>
<td>53</td>
</tr>
</tbody>
</table>

$R_{BP} = 8.20$ per SSVM execution or $R_{BP} = 1477$ per year summarizing all executions within a year.

However, the computation of the overall risk is only meaningful, if it is done for several BP in order to compare them. The multitude of processes can either be several different processes or it can be the same process in different variations.

3.5 Conclusions

The presented BPM-Risk analysis approach focuses on the IS - rather then on the IT network behavior. As a consequence, critical (business) activities are identified instead of critical hardware, software, etc.

The BPM-Risk analysis approach is considered as a simplified risk ranking and prioritisation tool which cannot replace more sophisticated and established system modeling techniques. But for all that, the approach shows practical outcomes within the modeling assumptions and simplifications. It has the advantage that each company has knowledge about how their major BPs are operated. The generation of simple BPD usually is trouble-free. The quantification of BPM-Risk analysis parameters is still challenging. However, estimation is supported by simple categorizing lists or pure expert judgment.

In order to reduce the operating expense of risk analysts and to increase the level of detail and accuracy of the model, it is necessary to set up a more sophisticated model, e.g., a simulation model of the extended BPM. As a simulation technique, a discrete event algorithm can be applied. To simplify the transcription from a graphical BPD to the simulation model, an object library may be created (e.g., an activity object, an event object and a transition object) and connected by drag and drop. The failure of each activity can be modeled either stochastically or bases on an underlying state model of the used IT network. The latter enables to model interdependencies among the BPM activities.
As argued in the introduction of this Thesis, the ABM approach is regarded as a promising toolbox for modeling and simulation of complex, interdependent systems and at the same time, it promises to be convenient to simulate parallel BP. To explore the capabilities of ABM for this purpose, the next chapter addresses the feasibility of ABM in risk and reliability analysis.
Chapter 4

Agent-based Modeling for Risk Analysis

4.1 Agent-based Modeling

Agent-based modeling (ABM) is an upcoming approach in complex systems science to model structures comprising autonomous and interacting elements. Some scientists even denote this computer simulation based modeling approach "A New Kind of Science" [83] and argue, that besides deduction and induction, ABM and simulation is a third way of doing science.

The rapid development of computational power within the last decades enabled the application of ABM in various fields of application and the modeling of larger systems. The applications comprise traffic modeling [84], conflict research [85] or the modeling of stock markets [86] among others. As a literature research (see Section 4.1.3) clearly indicated, the ABM approach is surprisingly almost invisible till now in the risk and reliability research and publications. However, as argued in the previous chapters, this approach is seen as promising to combine with BP modeling techniques to establish a new and powerful tool for IS risk analysis. In order to introduce the ABM approach, this chapter gives a short introduction to the basic terms and definitions, followed by a literature research giving an overview of the use of ABM in risk research. To illustrate the application of ABM, a case study of a dynamic virus propagation is introduced and to demonstrate the capabilities and the correctness of ABM two further case studies are presented. All simulations within this thesis are implemented with AnyLogic [118].
4.1.1 Terms and Definitions

The basic idea of ABM is to model only the units - called agents - of a specific system and to simulate their interplay in order to derive and analyze the total system behavior. Although there is no standardized definition of the term agent, and the distinction between the terms agent and object is often not explicit, definitions tend to agree on more points than they disagree. The concept of agents is closely related to the object oriented approach used in modern programming languages such as Java or C++. Thereby an object is defined by its states and behavior (see Def. 8), whereas agents can be seen as objects with more extended capabilities (see Def. 9), e.g., rules of behavior, autonomy, cooperation (e.g., perception, action, communication), mobility, memory, learning ability, etc. In order to make a distinction between agents and objects it is suggested in literature [27, 28] that an agent should at least be characterized by possessing the properties of autonomy and cooperation, or attributes like adaptive behavior [88], i.e. the capability of an agent to learn and to adapt its behavior accordingly. To be able to adapt the behavior, agents need besides their basic decision rules a set of higher-level rules to change the rules [89]. The basic rules then represent the interaction with the environment and the higher-level rules are the basis for adaption. Many ABM are based upon a grid. Boccara [17] for instance ranks cellular automata among ABM, but in general, ABM are able to describe sparse situations with different types of agents.

However, this thesis is following a broad-minded definition of the term agent, which includes most of the existing definitions:

**Def. 8** Object: defined by its identity, state and behavior [90].

Identity is the property of an object that distinguishes it from other objects (e.g., the label safety valve). State describes the data stored in the object (e.g., open, closed) and behavior specifies the methods (functions) in the object’s interface by which the object can be used (e.g., valve opens when a specified threshold pressure is exceeded).

According to [91] the general definition of the term agent provided by [92] becomes increasingly accepted. It considers an agent as an element which has a set of design objectives and is capable to autonomously perform flexible actions in a dynamic and unpredictable environment in order to meet its goals. To even reduce the set of conditions, the following definition is used:

**Def. 9** Agent: object with extended capabilities.
These capabilities embrace rules of behavior, autonomy, cooperation, mobility, memory, learning abilities, among others (Fig. 4.1). Cooperation, which is considered as a core capability of an agent, comprises, e.g., perception and action (interaction) and communication.

![Diagram of agent properties]

**Figure 4.1: Agent properties**

Besides the capability of interaction, cooperation implies the presence of an environment $\Omega_i$ to cooperate with. Assuming that the set of all agents of a specific system is given by $\Sigma = \{agent_1, \ldots, agent_n\}$, the environment of $agent_i$ is defined as $\Omega_i = \Sigma \setminus \{agent_i\}$.

### 4.1.2 History of ABM

ABM has its historical roots in the 1940s in the science of Complex Adaptive Systems (CAS). The basic approach was to build models from the bottom-up instead of doing classical top-down modeling. The systems modeled in this way showed surprisingly complex top-level behavior like self-similarity, emergence or self-organization. One of the first example model which was heavily discussed among scientists was the *Game of Life*, a Cellular Automata (CA) presented by the British mathematician John Horton Conway in 1970 [93].

CAs were originally developed by Stanislaw Ulam to solve a mathematical problem formulated by John von Neumann [89]: The question was, whether it is possible to design a machine which is able to make an exact copy of itself. In other words: Is it possible to develop a logical structure which contains all the instructions for self replication? It turned out that the answer is yes. The proof was given by the abstract formulation of a machine in the form of a CA.
CAs are typically composed of cells on a two-dimensional grid. They change their properties (e.g., color) every finite time step following simple rules of behavior. The future state of each cell depends on the actual state of its neighboring cells.

This relatively simple arrangement has open up a paradigm shift in mathematics. Especially Stephan Wolfram formed the term CA and further developed the underlying theory and techniques. The title of his book "A New Kind of Science" [83] demonstrates what he is thinking of the possibilities of these approaches.

When scanning literature for ABM and related topics, it becomes clear that since Conway's *Game of Life*, many similar scientific trends have emerged in this fields which are sometimes hard to distinguish, but it is not the focus of this chapter to differentiate between all these approaches.

### 4.1.3 ABM in Risk Research

Where ABM is a well established concept in several fields (e.g., conflict research, urban planning, traffic modelling), it is not wide spread in risk assessment. An extended literature search gives information about the state of the art of ABM in the area of risk, reliability and hazard analysis as defined, e.g., in [11] and [94]. The literature is mainly extracted from [95–97], covering the core literature of the engineering field. Filter criteria are (a) publication year from 2000, (b) the methods described are clearly associated with risk assessment, (c) the methods described are allocatable to the principles of Definitions 8 and 9. However, the differentiation of the terms agent-based and object-oriented is often ambiguous in papers. It is not transparent to the reader whether the phrase "object-oriented" is used in strict terms of software development or it simply stands for modularisation. Three definitely object oriented and agent-based approaches to risk assessment are given in [98–100], [25, 80] and [101]. The latter methodology "integrates decision-theoretic troubleshooting within risk assessment for industrial process control" and advice on sequence of repair by probability-cost estimation is given. This methodology uses generic object-oriented Bayesian networks.

The methodology and tool introduced in [102] is a computer-aided prototype which "automatically derives different representations of fault propagation: fault trees, fault event trees and a diagnostic rules table" for industrial processes. The methodology expressly follows an "object-oriented modelling paradigm". The term agent is not used. Another object-oriented approach gives [103] in order to automatically create an FMEA (Failure Modes and Effects Analysis) diagram. There, entities (objects in accordance with Definition 1) are "able to act or respond to the system through its distinctive 'memory' "
which follows Definition 9. An older paper from 1996 [104] follows the same paradigm to construct a knowledge based system for Hazard and Operability Studies (HAZOP). Although not explicitly mentioned, the specified interplays among compound objects (chemical compounds, energy sources, environmental conditions) likely follow the definition of an agent.

There are a few papers dealing with ABM of maintenance management and scheduling, e.g., [105, 106]. However, the results rather comprise the analysis and optimisation of maintenance processes and do not provide any probabilities and other risk figures.

A number of U.S. organisations (both commercial and governmental) are involved in the development of new models for analysing critical infrastructure interdependencies also related to risk assessment aspects. An extended list of such projects is shown in [107]. Some of these approaches comprise ABM but astonishingly, there is almost no scientific publication visible about their achievements. Furthermore, these developments mainly result in the design of new and very complex simulation tools which makes it difficult to verify their results and conclusions.

Above all, some researchers in the field of financial risk assessment have discovered the promising potential of ABM to model and forecast the behavior of financial markets [86, 108–110]. Another example of an achievement related to safety analysis is [29]: The author investigated a situation of escape panic in an overcrowded area with limited exits. Using an agent based simulation model of human behavior, it was found that placing an obstacle in front of an emergency exit can increase the efficiency of such an exit due to unexpected selforganisation.

4.2 Open Questions

Engineered systems have witnessed greater and higher integration and increasing complexity while risk and reliability aspects have become increasingly important. Traditional analytical methods, such as FTA, following a static approach of decomposition and describing the system’s behavior by ‘the sum of its parts’, turned out to be inadequate to model and analyze such large, complex systems. ABM is often put forward as a promising approach. Surprisingly, ABM is reluctantly used in the area of risk and reliability as a literature review (see section 4.1.3) clearly indicated.

Therefore, it has to be clarified, that ABM is able to cover the main issues and challenges in analyzing risks of complex systems. As argued in the introduction section, traditional methods have difficulties to tackle the following problems:
• Risk number computation: The quantitative analysis of frequencies and consequences in complex systems is mostly goes beyond the traditional methods.

• Process modeling and interaction between humans and machines (interdependencies) is very challenging.

• Many complex and networked systems has a tendency for cascading failures. But again, established risk analysis methods can not provide reliable results.

In order to check, whether ABM is able to cover all these problems, three case studies were carried out: An agent-based virus propagation model was designed to examine the computation of frequencies and consequences for risk analysis (Case Study 1). In a second case study, different maintenance processes of operators were successfully analyzed within an ABM and in a third study, an ABM of a generic complex network was developed in order to demonstrate the ability of ABM to analyze event cascades in complex systems. All models used Monte Carlo simulation techniques for quantification.

4.3 Case Study 1: Epidemic Propagation of Diseases

The benefit of sophisticated models of disease propagation is undisputable. The relation to risk analysis can be seen in the field of probabilistic consequence and risk assessment. Further more, the models can be used to test countermeasures regarding their effectiveness [113–116] or to make forecasts of epidemics [117].

Epidemic propagation of diseases has been studied for a long time. Most classical mathematical models for the spread of disease use differential equations based on uniform mixing assumptions. These models are said to understate the role of non-homogenous mixing in populations with geographical and social structure [111]. In the latest years, a wide variety of spatial and network models have been proposed that incorporate various aspects of interaction structure among individuals, e.g., [112, 113]. All these new approaches rely on massive computer simulation. Most authors in this field of research do not use the term "agent-based model" what illustrates the unsatisfactory spread of this unique concept. In order to illustrate the principles of ABM and to demonstrate how it can be used for risk analysis, this section highlights an example of an agent-based virus propagation model.
4.3.1 Virus Propagation Model

Let’s assume that 500 individuals (agents) are altogether located on a square field of 400 x 400 meters (see Fig. 4.3). The agents, who represent people for instance, walk on a random path across this field. Therefore, the population is randomly distributed on the square. The agents cannot escape from this area. Each agent has four possible states. In the beginning, it is susceptible to a threatening infectious disease (e.g., a virus). Therefore, the initial state of the agents is called susceptible. If an agent meets an infected individual, it can become infected too, what leads to a state transition form the susceptible state to the infected state (see Fig. 4.2). During the time of sickness (infected state), the agent can infect other individuals by contacting randomly one of its nearest neighbors. For this purpose, the infected agents randomly perform an internal state transition (infected → infected). This state transition is performed with a contact rate $\lambda_c$ and activates an infection algorithm which first chooses randomly one of the neighbor agents of its environment $\Omega_i$ within a specified radius $R_c$ of, e.g., 5 meters (see Fig. 4.3) and then sends a signal event to this close individual.

![Agents state-chart](image)

**Figure 4.2: Agents state-chart**

This signal causes another state transition at the chosen agent from susceptible to infected if this individual is still susceptible at this time. Otherwise, nothing happens, and the signal expires. The signal event is an abstraction of an infectious event, e.g., coughing and splashing contaminated saliva to the neighbor. With the rate of $\lambda_{incub}$ (equal $\frac{1}{incubationtime}$) the infected agents change their state again. With the probability $p_{mort}$, they reach the final state "dead" and with the probability $1 - p_{mort}$, they recover and reach the state recovered. If an agent has reached this state, it is resistant against future infections.

It is well known, that the spread of diseases is considerably influenced by the mobility
of the individuals of an affected population. High mobility causes a higher probability of epidemics. In order to test this rule, the presented virus propagation model is extended:

To describe the mobility of the moving agents, they are designed performing a random walk. This movement is defined in Equations 4.1 to 4.4. In order to be able to investigate the influence of the agents mobility, the mobility parameter $m$ can be varied, where a high value of $m$ represents a highly mobile population and a low value of $m$ describes a rather inactive population.

$$\Delta v_x(t_i) = \text{uniform}(-m, m)$$
$$\Delta v_y(t_i) = \text{uniform}(-m, m)$$

$$\vec{v}(t_i) = \begin{pmatrix} v_x(t_i) \\ v_y(t_i) \end{pmatrix}, \quad \Delta \vec{v}(t_i) = \begin{pmatrix} \Delta v_x(t_i) \\ \Delta v_y(t_i) \end{pmatrix}, \quad \vec{x}(t_i) = \begin{pmatrix} x(t_i) \\ y(t_i) \end{pmatrix}$$ (4.2)

$$\vec{v}(t_{i+1}) = \vec{v}(t_i) + \Delta \vec{v}(t_i)$$ (4.3)

$$\vec{x}(t_{i+1}) = \vec{x}(t_i) + \vec{v}(t_{i+1})$$ (4.4)

### 4.3.2 Simulation of an Epidemic

During a simulation of the presented model, global variables of the system are observed, e.g., the total number of fatalities or the number of infected, recovered or still susceptible people. The evolution of these numbers during a model run (MR) is exemplified in Fig. 4.4. The simulation is stopped when there exists no infected agent.
4.3. Case Study 1: Epidemic Propagation of Diseases

anymore. In the presented example, the model results in 28 fatalities, 361 recovered and 111 still susceptible individuals.

![Figure 4.4: Characteristic devolution of an epidemic.](image)

Running the simulation model several hundred times with the same input parameters, it is possible to record a histogram approximating the probability distribution of these observables (see Fig. 4.5).

In order to investigate the influence of model parameters such as the mobility $m$ on the resulting distributions, the corresponding parameter is varied, and for each setting, the distributions are computed. An example is displayed in Fig. 4.5. The right graph (C) of this Figure represents for instance the results of 2712 simulations with $m = 4$ (high mobility). In this case, most of the simulations (73%) showed an epidemical spread of the disease resulting in more than 25 fatalities. Although, 24% of the simulations didn’t show a cascading behavior at all (less than 6 fatalities), which is an interesting characteristics of epidemics. This means the first phase, e.g., when a new virus appears, is very important. If the initially infected individual does not infect a minimum threshold of other individuals, there will be no epidemic.

To quantify and evaluate the influence of the mobility on the epidemic spread of a disease, risk concepts can be applied. To visualize the impact of a mobile population, the data presented in Fig. 4.5 is summarized in a cumulative mortality diagram (see Fig. 4.6, left graph). To rate the different scenarios, the definition of risk according to chapter 2 of this thesis is used, where $C_i$ represents the number of fatalities. If the mobility $m = 0$, what indicates that the agents do not move, the expectancy value of the total number of fatalities ($= Risk$) is about 4.2. With $m = 2$, this value rises to 14.8 and with $m = 4$, the expected number of fatalities reaches 29.2. This experiment already leads to an important conclusion: If you want to prevent a threatening epidemic, tell
people to stay at home and advise them, not to move. In 2003, the Chinese government demonstrated the effectiveness of such a measure by imposing a country wide curfew to confine the spread of the SARS disease.

As risk number, the expectancy value of the amount of fatalities can be chosen. The effectiveness of countermeasures, such as preventive vaccination, a reduction of people’s mobility, or use of quarantines etc. can be simulated and then be rated with a risk number.

### 4.4 Case Study 2: Maintenance Processes

A field of using the ABM approach in risk assessment is the availability computation of complex systems. In order to illustrate the extended capabilities of ABM, a system of $n$ identical and independently operated units will be analyzed by means of Markov chains, Monte Carlo simulation and ABM. The unit states are Boolean (operating/non-operating).

#### 4.4.1 Markov Model

In a Markov state diagram, a transition from an operating state to a non-operating state of a unit is indicated by a failure rate, while the repair rate quantifies a recovery action. Eq. 4.5 shows the stationary probability $p$ of a unit to be in the state "operating", where MTTF (Mean Time To Failure) and MTTR (Mean Time To Repair) are the reciprocals of

![Markov Model Diagram](image_url)

**Figure 4.5:** Distribution of modeled fatalities caused by an epidemic. $p_{\text{mort}} = 0.1$, $\lambda_c = 5 \text{ week}^{-1}$, $\lambda_{\text{incub}} = 1 \text{ week}^{-1}$, $R_c = 25m$, $N = 500$, $\Delta t = 0.5 \text{ week}$, and (A) $MR = 9972$, $m = 0$, (B) $MR = 3501$, $m = 2$, (C) $MR = 2712$, $m = 4$ ($MR = \text{Model Runs}$)
4.4. Case Study 2: Maintenance Processes

Figure 4.6: Cumulative mortality per event for the three scenarios A, B, and C (left graph) and the influence of the mobility parameter \( m \) on the risk (right graph).

The associated rates \( \lambda \) and \( \mu \).

\[
\lim_{t \to \infty} p(t) = \frac{\mu}{\mu + \lambda} = \frac{MTTF}{MTTF + MTTR} = \rho
\]  

(4.5)

The Markov state probabilities \( p_i \) are given by the Binomial distribution (Eq. 4.6) assuming independent failure- and repair events (i.e., the number of operators \( N_O \) doing repair work is the same as the number of units \( n \Rightarrow N_O = n \), consequently the repair action starts immediately after the failure event). Then, \( p_i \) is the probability of finding \( i \) operating units out of total \( n \) units with an expected value \( m \) (Eq. 4.7) and standard deviation \( \sigma \) (Eq. 4.8).

\[
p_i = \binom{n}{i} \cdot p^i \cdot (1 - p)^{n-i}
\]  

(4.6)

\[
m = n \cdot p
\]  

(4.7)

\[
\sigma = \sqrt{n \cdot p \cdot (1 - p)}
\]  

(4.8)

This simple model becomes more complicated if the assumption of independence between failure- and repair rates is abandoned. Assuming, for instance, a limited number \( N_O \) of operators (i.e. \( N_O < n \)), the probability \( p_i \) (equivalent to \( p_i \), Eq. 4.6) of finding \( i \) out of \( n \) operating units is given by Eq. 4.9 and Eq. 4.10 (where \( \rho = \frac{\lambda}{\mu} \)), while the mean number of operating units \( m(N_O) \) is computed by Eq. 4.11 and the steady state probability \( p(N_O) \) (equivalent to \( p \) in Eq. 4.5) of finding a specified unit in the operating state (availability) is derived from Eq. 4.12.
\[ \pi_0 = \left[ \sum_{i=0}^{N_O} \binom{n}{i} \rho^i + \sum_{i=N_O+1}^{n} \frac{n!}{N_O^{-N_O} N_O!(n-i)!} \rho^i \right]^{-1} \] (4.9)

\[ \pi_i = \begin{cases} \binom{n}{i} \rho^i \pi_0 & 1 \leq i \leq N_O \\ \frac{n!}{N_O^{-N_O} N_O!(n-i)!} \rho^i \pi_0 & N_O \leq i \leq n \end{cases} \] (4.10)

\[ m(N_O) = \sum_{i=1}^{n} \pi_i \cdot i \] (4.11)

\[ p(N_O) = \frac{m(N_O)}{n} \] (4.12)

Both the previous (Eq. 4.5 to 4.8) and the extended model (Eq. 4.9 to 4.12) show the same results, if the values of \( \lambda \) and \( \mu \) are kept the same and if there is an operator \( j \) available for each unit \( i \). This can be illustrated, e.g., by the relative availability \( \frac{p(N_O)}{p} = 1 \) \( \forall N_O \geq n \). But in case the number of operators is reduced one by one, it can occur that non-operating units have to wait for an operator. This reduces the steady state probability \( p(N_O) \) of finding a machine in the operating state (Fig. 4.7).

![Figure 4.7: Relative availability \( \frac{p(N_O)}{p} \) of a single unit in a n-unit system, where \( n = 50 \) and \( \rho = \frac{\lambda}{\mu} = 1 \), i.e. \( p = 0.5 \) (see Eq. 4.5).](image)

### 4.4.2 Monte Carlo Simulation

The same Markov model can be solved by a Monte Carlo simulation whereby the two-state model of a unit is set up as a simulation experiment. Again, the number of operators \( N_O \) is equal to \( n \). This needs the creation of a simple object which is represented
4.4. Case Study 2: Maintenance Processes

by the state machine (Fig. 4.8) of a unit. The sampling values of the random variables TTF (Time To Failure) and TTR (Time To Repair) are exponentially distributed and are generated by a random number generator (Eq. 4.13 and 4.14),

\[ TTF = -MTTF \cdot \ln(U) \]  \hspace{1cm} (4.13)

\[ TTR = -MTTR \cdot \ln(U') \]  \hspace{1cm} (4.14)

where \( U \) and \( U' \) are uniformly distributed random number sequences between 0 and 1. If a state machine performs a transition, the next random time is generated and assigned to the corresponding state machine. So each unit has a state function \( S_{M,i}(t) \) indicating its state (0,1) at time \( t \) (Eq. 4.15).

\[ S_{M,i}(t) = \begin{cases} 1 & \text{working at time } t \\ 0 & \text{else} \end{cases} \]  \hspace{1cm} (4.15)

\[ w(t) = \sum_{i=1}^{n} S_{M,i}(t) \]  \hspace{1cm} (4.16)

The mean number of available units is derived from \( w(t) \) by sampling the curve within time-intervals of \( \Delta t \). This leads to a set of \( N+1 \) samples \( w_k = w(k \cdot \Delta t) \) for \( k = 0, \ldots, N \). \( N \) is an integer depending on the modeled time \( t \) and the sample interval \( \Delta t \), i.e.,

\[ \frac{t}{\Delta t} - 1 \leq N \leq \frac{t}{\Delta t}, \forall N \in \mathbb{N} \]  \hspace{1cm} (4.17)
This model extension serves as a first step in building up a sophisticated representation of the real technical systems and their processes.

### 4.4.3 Cut-off Criteria

The computed average number of units $m_w$ finally converges to the analytical mean value $m$ (Eq. 4.7) as the simulation proceeds. The simulation is terminated when $m_w$ reaches a specified confidence level. In order to define a stopping rule, the standard deviation of $w_k$ is used:

$$
\sigma_w = \sqrt{\frac{1}{N+1} \sum_{k=0}^{N} (w_k - m_w)^2}
$$

The confidence level of $m_w$ is assessed by $\sigma_m$:

$$
\sigma_m = \frac{\sigma_w}{\sqrt{N}}
$$

The simulation is stopped when $\frac{\sigma_m}{m_w} < \alpha$, where $\alpha$ is the confidence level of the result. The best estimate $p_w$ of the probability $p$ is given by Eq. 4.21 which represents the steady state probability of a unit being in the "operating" state.

$$
p_w = \frac{m_w}{n}
$$

In order to compare the simulation results with the analytical Markov solution, the relative availability $\frac{p_w}{p}$ is used. It is obvious, that for this Monte Carlo simulation $\lim_{t \to \infty} \sigma_m = 0$ and $\lim_{t \to \infty} \frac{p_w}{p} = 1$, which demonstrates the equivalence to the analytical solution (Eq. 4.5 to 4.7).

### 4.4.4 Agent-based Model

A limited number of operators (agents) $N_O$ is added to the model in order to get a more realistic consideration of repair actions. The transition of a state machine from operation
to non-operation indicates the need for an operator. The unit randomly chooses an idle operator and sends the message "repair me". If there is no idle operator, no massage is sent.

![Figure 4.9: Agent-based model of the units.](image)

A unit is repaired (no delay for travels) by the chosen operator. At the same time, the operator agent performs a transition from "idle" to "repairing". This transition is triggered by a message. A random timer generates $TTR$ according to Eq. 4.14. On average, an operator needs the $MTTR$ for a repair action. It is assumed, that every triggered repair-action is completed successfully. When the timer expires, the message "repaired" is sent to the unit $i$ which causes a state transition from the "not operating" to the "operating" state. The operators' new state now depends on the amount of non-operating units (Fig. 4.10).

![Figure 4.10: Agent-based model of the operators.](image)

If another unit is to be repaired (i.e., $n_{UTR}(t) > 0$), the operator $j$ randomly chooses and repairs one of these units. To indicate that this unit is now beeing treated, the operator
sends to it the message "haveYou". In this case, the operator remains in the same state ("repairing") after the state transition. In case of $n_{MTR} = 0$, the operator changes to the "idle" state. The state function of the operator is given by Eq. 4.22.

$$S_{O,j}(t) = \begin{cases} 
1 & \text{idle at time } t \\
0 & \text{else}
\end{cases}$$ (4.22)

The number of available operators $a(t)$ is computed analogously to Eq. 4.16,

$$a(t) = \sum_{j=1}^{N_O} S_{O,j}(t)$$ (4.23)

and the number of units to be repaired at the time $t$ can be derived by

$$n_{UTR}(t) = n - w(t) - (N_O - a(t) + 1)$$ (4.24)

Where $n - w(t)$ counts the non-operating units and $N_O - a(t) + 1$ the repairing operators (+1 because the corresponding operator is still in the state ("repairing" although he is free now).

Compared to the Monte Carlo simulation presented in section 4.4.2, this modeling extension adds agent properties to the units and introduces further agents (operators) to the system. Messages and message transitions now bring in the capability of cooperation among the agents. The environment view $\Omega_i$ (i.e., operators) is also added to the system to be modeled.

### 4.4.5 Extending the Model

In order to make the model even more realistic, new features are integrated. A next step is to consider the path of the operators, e.g., from the home office to the units and vice versa, resulting in an additional delay time $TTT$ (Time To Target). Therefore the operators (and the units) have to be extended by featuring their location $(x, y)$ (Fig. 4.11) and adapting their behavior to the new situation (Fig. 4.12).

In order to keep the model as simple as possible, the units with the coordinates $(X_i, Y_i)$ are placed in a circle $X_i^2 + Y_i^2 = R^2$ and the home office in the center, so the distance from the home office to each unit is given by $R$ (Fig. 4.11). If necessary, it is possible to introduce any distribution of the units, e.g., with different offices on different floors in various buildings. As the agents (operators and units) have now the spatial property,
the operators need the new state "on the way" to represent their behavior. If a unit \( i \) fails, it sends the message "repair me" to one of the "idle" operators \( j \). This message triggers a state transition from "idle" to "on the way" in the state-chart of operators \( j \).

For leaving the state "on the way", the operators position \( \vec{x}_j = (x_j, y_j) \) has to be equal to the target position \( \vec{x}_{tg} \) of the operator \( (\vec{x}_{tg} = (X_i, Y_i) \) if unit \( i \) is the target of operator \( j \) and \( \vec{x}_{tg} = (0, 0) \) if the operator returns to the home office). For this purpose, the operator needs to change his position and moves with the velocity \( \vec{v} = (v_x, v_y) \) to the target position. Within the simulation, this movement is modeled by \( \vec{x}_j(t + \Delta t) = \vec{x}_j(t) + \Delta t \cdot \vec{v} \), where \( \Delta t = \frac{1}{|\vec{v}|} \) is a finite time-step and \( |\vec{v}| \) is the norm of the operator's velocity vector.

As discussed in the next chapter, the maintenance strategy of the operators (e.g., how to choose the next target position \( x_{tg} \), when to go back to the home office [condition 1] and when to go to repair another unit [condition 2]) has an essential influence on the model result.
4.4.6 Evaluating Maintenance Strategies

For this case study, five basic maintenance strategies were analyzed, which add up to the formal and illustrative examples of Table 4.1. These strategies can be composed to complex processes. In all cases it is assumed that the operators in the state "on the way" are unavailable for other units and thus they cannot change their target on the fly. Strategies S0 to S3 are self-explanatory. S4 is only reasonable in case of high unit failure rates.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>An operator needs no time to go to a failed unit (idealized strategy).</td>
</tr>
<tr>
<td>S1</td>
<td>An operator always goes back to the home office after he has repaired a unit.</td>
</tr>
<tr>
<td>S2</td>
<td>After the operator has finished a repair action, he chooses the nearest unit failed at right-hand. If there is no unit to repair, he goes back to the home office.</td>
</tr>
<tr>
<td>S3</td>
<td>After the operator has finished a repair action, he checks whether one of the two neighbor units has to be repaired. If so, he goes to one of these units, otherwise he goes back to the home office.</td>
</tr>
<tr>
<td>S4</td>
<td>The operator never goes back to the home office. After he has finished a repair action, he checks whether one of the two neighbor units has to be repaired. If not, he stays where he is.</td>
</tr>
</tbody>
</table>

4.4.7 Optimal Strategies

The different results (relative availabilities $\frac{p_w(N_{O})}{p}$) of the simulation with the five different strategies described in Table 4.1 are summarized in Fig. 4.13, where $p_w(N_{O})$ is computed according to $p_w$ (Eq. 4.21) for $N_{O} = 1 \ldots 70$. In both cases ($\frac{\mu}{\lambda} = 1$ and $\frac{\mu}{\lambda} = 10$), S0 turned out to be the best strategy as it always returns the highest value of $\frac{p_w(N_{O})}{p}$. But S0 is a virtual strategy, i.e. it results exactly in the same numerical values as the output of the analytical model (Eq. 4.9 to 4.12) without the spatial property (see Fig. 4.7). This is nothing exciting as this strategy assumes that the operators need no time to move.
from one place to the other \((v \to \infty)\), which makes it the same model as the one without spatial information. In the case where \(\frac{\mu}{\lambda} = 1\), S4 appears to be the best solution. In the other case, with \(\frac{\mu}{\lambda} = 10\), S2 is the best strategy for \(N_O < 20\) and again, S4 is the best solution for \(N_O \geq 20\). In this case, a combination of S2 and S4 seems to be optimal while strategies S1 and S3 never are.

**Figure 4.13:** Relative availability \(\frac{p(N_O)}{p}\) of a single unit in a \(n\)-unit system maintained by different strategies \((S0 \ldots S4)\), where \(n = 50\) and \(\frac{\mu}{\lambda} = 1\) (left graph) and \(\frac{\mu}{\lambda} = 10\) (right graph).

### 4.4.8 Emergent System Properties

The most interesting outcome of this simulation experiments is the output of S2 for \(\frac{\mu}{\lambda} = 1\). It turns out, that there exists an optimal number of operators at \(N_O = 27\) for which the relative availabilities \(\frac{p_w(N_O)}{p}\) have a local maximum (Fig. 4.13, left graph). Adding more operators to the system, the relative availability \(\frac{p_w(N_O)}{p}\) decreases. This is a completely unexpected system behavior. It would be expected, that an increasing number of operators would shorten the mean time to repair because an additional operator can start repairing a failed unit even if all other operators are still occupied. This exemplifies the emergence of complex behavior shaped in a kind of phase-transition. This system property appears as a result of the interactions among a number of simple entities (units and operators) operating in a shared environment, showing a more complex behavior as a collective.

In the first phase \((N_O = 1 \ldots 27)\), the operators always choose the nearest not-operating unit on their right-hand side as next target. So far, the distance between the operators’ starting point and its target is usually less then \(R\) (radius of the unit circle). Consequently the Mean Time To Target \((MTTT)\) is less than \(R/|v|\).
In the final phase ($N_O > 60$), the system behaves exactly the same as with strategy S1, where the operators always go back to the "home office", because there is never an additional unit to repair and the operators always return to the "home office". Here, where the number of operators assures that there is at least one operator in the "home office", $TTT$ is constant ($MTTT = TTT = R/|v|$), because every unit is served from an operator from the "home office".

![Graph](image.png)

**Figure 4.14:** Relative mean time to target ($MTTT/|v|$) against the number of operators $N_O$, ($\frac{\mu}{\lambda} = 1$, $n=50$, Strategy S2).

In-between these two phases, the system experiences a phase transition, where $MTTT$ of the operators can reach a higher value, i.e. $R/|v| < MTTT < 2R/|v|$ (see Fig. 4.14). In this transition phase, the operators "handicap" each other, which decrease the efficiency of their work. This handicap is caused by the fact, that the distance to the nearest unit to repair can be larger than $R$ because other operators are already repairing closer units.

### 4.4.9 Cost-benefit Analysis

In order to use such ABM for sophisticated cost-benefit analysis, it is necessary to know the relation between the total salary of operators $N_O$ and the expected loss caused by a too low availability. In Fig. 4.15, an example of a cost-benefit analysis is presented. It is assumed that the expected loss (risk) of the system is proportional to the relative availability $\frac{p_{w}(N_O)}{p}$, e.g., $L \cdot (1 - \frac{p_{w}(N_O)}{p})$, where $L$ denotes a loss parameter. The maintenance costs are assumed to be proportional to the total salary, e.g., $C \cdot N_O/n$, where $C$ represents a parameter indicating the cost per operator.

For $L = C = 1$, the expected loss as well as the operators salaries are values of the
Figure 4.15: Cost-benefit analysis of the system \((n = 50, \mu = 10, L = C = 1)\), applying the maintenance strategies \(S_2 (N_0 < 20)\) and \(S (N_0 \geq 20)\), displaying the maintenance costs (dotted line), the expected loss (thin continuous line) and the sum of both (bold continuous line) left graph:. Optimal number of operators (same model parameters) for different fractions of \(L/C\) right graph:.

set \([0, \ldots, 1]\), i.e. maximum loss and maximum salary cost are normalized to 100%. If the system represents a supply system for any services, which are important for the value generation of a company, it can be calculated, how many operators should be employed in order to maximize the benefit and reduce the risk of not fulfilling service level agreements. The optimal number of operators does not only depend on \(n, \mu\) and \(\lambda\) but also on \(L/C\). If this ratio is small, the optimal number of operators is also small, because the potential loss is small compared to the operators’ cost. Else, when \(L/C\) is high, the optimal number of operators is much higher, i.e., it is worth to invest in more operators to prevent higher losses (see Fig. 4.15, right graph). This figure also highlights the phase-transition between \(S_2\) and \(S_4\) at \(L/C = 2.57\) where \(N_0\) jumps from 13 to 25.

### 4.5 Case Study 3: Cascading Failures in Complex Networks

Many complex systems comprise a networked structure, where an initial failure event may evolve into a cascading failure propagation. To investigate the capabilities of ABM to model such systems for risk analysis, a third study was performed. Therefore a generic model which considers two stress induced component outage types differing from each other in their timescale, was developed in cooperation with a team at the
4.5.1 Failure Propagation Model

The Nodal Stress Model. The components of a real technical network are represented by the vertices of an undirected, connected graph being further referred to as "nodes" and "network" respectively. The links of the network determine whether there is an influence between a pair of connected nodes. Within the scope of this case study a very simple model for the nodal stress and its propagation through the network is used. It is assumed, that the stress $s_i(t)$ on node $i$ as the ratio of the number of failed neighboring (i.e. initially connected) nodes to the total number of initially connected nodes. By measuring the internal state of the node by a boolean variable $x_i(t)$, where $x_i(t) = 0$ represents an operating node and $x_i(t) = 1$ a failed node, $s_i(t)$ is calculated as

$$s_i(t) = \frac{\sum_{j \in \Omega_i} x_j(t)}{k_i}$$

(4.25)

where $k_i$ is the nodal degree, i.e. the number of links from node $i$ to its initially connected nodes $j$ of the set $\Omega_i$. Eq. 4.25 implies the assumption that highly connected nodes have a stronger fault tolerance than nodes with lower connectivity. Furthermore the strength of the influence between two nodes is supposed to be the same for each pair of connected nodes.

The Concept of Critical Operating Temperature. The temperature - heat analogon is derived from the assumption, that the stress $s_i(t)$ influences the failure-probability of the given node by increasing its "operating temperature" $T_i(t)$ with a time delay due to thermal inertia effects.

$$\frac{dT_i(t)}{dt} = \frac{1}{a} s_i(t) - \frac{1}{\tau} T_i(t)$$

(4.26)

The stress $s_i(t)$ thus can be interpreted as the actual heat inflow, $a$ as the heat capacity of the node and $\frac{1}{\tau} T_i(t)$ as the temperature dependent heat outflow. In order to simplify Eq. 4.26, the temperature $T_i(t)$ is normalized with its maximum $T_{i,\text{max}} = \tau \frac{s_{i,\text{max}}}{a} = 1$. The maximum stress is $s_{i,\text{max}} = 1$ (see Eq. 4.25) resulting in $a = \tau$, where $\tau$ represents the characteristic time constant. The node fails when reaching its maximum or "critical"
operating temperature \( T_{i,\text{crit}} \) and therefore changes its state from \( x_i = 0 \) to \( x_i = 1 \). Although only nodes of the same type are considered, it is assumed that \( T_{i,\text{crit}} \in \{0, \ldots, 1\} \) are beta-distributed with the density-function \( f(T_{i,\text{crit}}) = \frac{1}{B(p,q)} T_{i,\text{crit}}^{p-1}(1 - T_{i,\text{crit}})^{q-1} \). The parameters \( p \) and \( q \) can be represented as functions of the mean value \( \langle T_{i,\text{crit}} \rangle \) and the variance \( \sigma^2 \) of the beta distribution.

**Wearout Model** The wearout model is based on the assumption, that the history of the stress level, as being reflected by the temperature sequence \( T_i(t) \), has a strong influence on wearout processes. This relationship between the stress and the long-term failure time of a component is widely used for accelerated component life testings on, e.g., electronic devices [119]. To include this kind of failure mode, a 0th order kinetics is adopted, where the "substance" \( A_i(t) \) of node \( i \) is decomposed by the process rate \( r_i(t) \):

\[
r_i(t) = -\frac{dA_i(t)}{dt} = cT_i(t) + r_{i,0}
\]  

(4.27)

The decomposition rate has a part being proportional to the actual operating temperature with \( c \) as a linear factor and a constant part \( r_{i,0} \) being independent from \( T_i(t) \) in order to account for spontaneous failures. Hereby, more sophisticated models for the temperature dependance of the wearout mechanism such as, e.g., the Arrhenius equation [9,119] where not used as this would require the estimation of even more model parameters. At \( t = 0 \) an initial amount or "contingent" \( A_{i,\text{crit}} \) is assigned to every node, so that the node fails if all of its contingent is decomposed, i.e. if \( A_i(t) = 0 \). Further assuming an exponential distribution for the temperature independent wearout failure times \( MTTF = A_{i,\text{crit}}/r_{i,0} \) the PDF of \( A_{i,\text{crit}} \) is calculated as \( f(A_{i,\text{crit}}) = \frac{1}{\langle A_{i,\text{crit}} \rangle} e^{-A_{i,\text{crit}}/\langle A_{i,\text{crit}} \rangle} \), with mean value \( \langle A_{i,\text{crit}} \rangle \).

### 4.5.2 Experimental Results

**Network Topologies** This failure propagation model was applied on two types of artificially created topologies: 1) random graphs and 2) scale-free networks [1], [120]. Both networks have \( N = 200 \) nodes and an average node degree of \( \langle k \rangle \approx 2.62 \). The scale-free networks have an algebraic distribution degree \( P(k) \sim k^{-2.5} \). The two network topologies were generated by a preferential attachment method [1]. For the scale-free network, the probability \( \Pi(k) \) that a new node is connected to node \( i \) is proportional to the degree \( k \) of node \( i \). To ensure that the network represents a connected graph,
every new node is attached to at least one node of the network. The generation of the random network was realized in the same manner by keeping the probability \( \Pi(k) \) the same for each node.

**The Spreading of Cascading Failures**  The results of two parameter variation studies are discussed in detail: 1) the sensitivity of the time dependent failure propagation with respect to the critical nodal operating temperature \( \langle T_{i,\text{crit}} \rangle \) and 2) the influence of the long-term wearout failure times which are varied by setting different \( \langle A_{i,\text{crit}} \rangle \). In the scope of this case study the results of the random network will not be discuss, as the influence of its topological difference to the scale-free network has already been analyzed by earlier work, e.g., [121, 122]. Concerning the temperature model (Eq. 4.26 a fixed variance of \( \sigma^2 = 0.0025 \) for the beta distribution of \( T_{i,\text{crit}} \) and a fixed characteristic time constant of \( \tau = 15 \) minutes were used for both experiments. The parameters of the wearout model (Eq. 4.27) were set to \( c = 1 \) and \( r_{i,0} = 0.1 \).

The results of the first experiment are shown in Fig. 4.17. The variation of the critical operating temperature \( \langle T_{i,\text{crit}} \rangle \) has a significant influence on the starting time of the fast cascade as well as on the gradient of the network decomposition. While for large \( \langle T_{i,\text{crit}} \rangle \) a quite slow and continuous regime (Fig. 4.17, continuous line) is observed, the fast cascading sequence for small \( \langle T_{i,\text{crit}} \rangle \) occurs rather early, at a higher ratio of intact nodes and at a significantly higher speed (Fig. 4.17, dotted line).

Within both failure propagation sequences the node outage due to reaching its critical

![Figure 4.16: Example of a generated random network (left side) and a scale-free network (right side)](image-url)
Figure 4.17: Failure sequence on scale-free networks with $MTTF = 1$ year for three values of $\langle T_{icrit} \rangle = 0.4, 0.5, 0.6$ (dotted, dashed and continuous line, respectively).

temperature clearly dominates the outage due to wearout within the fast cascading regime, as its characteristic time constant $\tau$ is in order of magnitudes smaller than the wearout failure times (Fig. 4.18).

Figure 4.18: Individual failure modes corresponding to Fig. 4.17, whereas the three larger slopes represent the failures due to reaching $T_{icrit}$ and the flat slopes the outages due to wearout.

The results of the second experiment is depicted in Fig. 4.19, where the sequence of the averaged nodal failure rate $\lambda(t) = -\frac{1}{N(t)} \frac{dN(t)}{dt}$ for a fixed $\langle T_{icrit} \rangle = 0.5$ and different $MTTF$ is plotted. Similar to $\langle T_{icrit} \rangle$ the $MTTF$ has a strong influence on the time of
occurrence of the fast cascading regime. While at the beginning of this experiments the failure rate is small and slowly increasing, it rapidly rises to a sharp peak when it comes to the fast failure propagation process.

Figure 4.19: The failure-rate $\lambda(t)$ during a cascade in a scale-free network for $\langle T_{i,crit} \rangle = 0.5$ and different MTTF, i.e. 1 week, 3 months, 6 months and 1 year (triangles, squares, diamonds and circles respectively)

4.6 Lessons Learned

ABM turned out to be applicable and to deliver correct results, i.e. it is comparable with the other modeling approach for same conditions. Moreover, ABM models have been successfully extended to include, e.g., various maintenance strategies and have demonstrated their capability to cope with dynamic system behavior and detect emergent system properties.

A first attempt has been made to use ABM for more than purely risk and reliability analysis, i.e. for cost-benefit analysis. It turned out that ABM - normally combined with other methods and simulation techniques (hybrid modeling approach) - has a high potential to help to realistically model large and complex systems even with limited, justifiable mathematical efforts, proper verification of the models and results provided.
Chapter 5

Agent-based Modeling of Business Processes

Chapter 5 aims to define a consistent transformation from process-oriented, semi-formal models to object-oriented formal simulation models. In a further step, the simulation model will be tuned for risk-number generation. For the implementation of the simulation code, a software called AnyLogic™ [137], a Java based simulation tool, is used.

5.1 Agent-based View of Processes

Commonly, a business process (BP) is seen as a set of one or more linkend procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships [74]. Most of the coherent representations, i.e. business process model (BPM), display BP in a process view, that is a chain of activities and / or events.

In order to group the individual processes, e.g. according to their stakeholders, the business process modeling notation (BPMN) provides a grouping element called swimlanes (see chapter 3). Using these swimlane elements, it is possible to identify all the agents involved in a BP. The mapping of a complex BP to its agents (stakeholders) yields in an agent view of the BP (Figure 5.1).

This agent view of a process is used in order to deduce an appropriate meta-model for the agent based simulation of BP. Then BP charts are transformed into the formal framework of a simulation. This step is standardised by the use of the "Business Pro-
Figure 5.1: Agent view of the BP representation of a generic IS

5.2 Basic Meta-Model

The analysis of the abstract formulation of a BP, its impact and dependency on, e.g. the physical equipment or the involved staff, leads to the conclusion, that a meta-model designed for the realistic computation and simulation of a BP needs to include two kind of components: Process components (abstract) and physical components (see Figure 5.2).

Process Components: A company’s BP is specified as a quadrupel of objects $BP = \{E, A, F, G\}$ where an event $E$ is something that happens during the course of a BP. An activity $A$ is a generic term for work that a company performs, a flow $F$ is used to show the order in which activities are performed in a process, and gateways $G$ are used to control the divergence and convergence of a flow $F$ [46]. A linear BP is a sequence of activities which is initiated by a start event and terminated by an end event (intermediate events are not necessarily required). Gateways enable the representation
of a branched BP which may end in many events. The activity sequences and events (vertices) are linked by flows (edges), i.e., a BP is represented as a (directed) graph.

**Physical Components:** The activities described in a BP mostly depend on real people (e.g., a companies’ staff) and real physical material (e.g., computers, networks). In the presented meta-model, these kind of process-participants are referred to as *physical objects*. These "objects" have a strong influence on the BP performance. If an activity, e.g., requires the availability of a specified PC, a network and a computer server, the outage of one of these objects makes it impossible to execute the corresponding activity successfully at a given time. On the other hand, the BP may contain a maintenance- or recovery-process for each failed physical object. For instance the inclusion of gateways (e.g., AND_Split, AND_Combine, XOR_Split according to the BPMN) enables the analyst to represent both successful and failed activities resulting in different end states. An activity’s failure is taken into consideration either stochastically or by the underlying (physical object) state model.

**Virtual Communication System:** The designed meta-model has to account for the interdependencies mentioned above. Therefore a "virtual communication system" for the communication between the process components and the physical components is designed. This communication system is used only for simulation purpose to enable appropriate interactions among these two kind of components. The "virtual communication system" mainly consists of "Tokens" and "Drones" which are message carriers, and a "Virtual Token Router" (VTR) component (see section 5.2.3.4), which handles the routing of "Tokens" and "Drones".

![Figure 5.2: Components of the meta-model](image-url)
Herewith, the proposed BPM risk analysis approach is a two-stage procedure: First, a conventional BPM (e.g., flow chart BPMN) is built up; second, physical objects are selected and linked.

### 5.2.1 Process Components

A BP is represented by a chain of (at least one) start event, activities, and end events. In order to simulate the state and progress of a BP, the virtual object "Token" is defined (see section 5.2.3.1).

#### 5.2.1.1 Start Event

The start event is designed as a token generator which is operated in a stochastic or signal mode. In the stochastic mode, the start event releases tokens in a stochastic time-interval $t_{\text{Token,mode}}(t)$. Two stochastic modes are implemented: One with exponentially distributed inter-arrival times (Eq. 5.1) and one using the triangle distribution (Eq. 5.2) for the random generation of token inter-arrival times:

$$t_{\text{Token,exp}} = -\frac{1}{\lambda} \cdot \ln(U')$$  \hspace{1cm} (5.1)

$$t_{\text{Token,tri}} = \begin{cases} 
\alpha + \sqrt{U(\beta - \alpha)(\gamma - \alpha)} & 0 \leq U \leq \frac{\gamma - \alpha}{\beta - \alpha} \\
\beta - \sqrt{(1 - U)(\beta - \alpha)(\beta - \gamma)} & \frac{\gamma - \alpha}{\beta - \alpha} < U \leq 1 
\end{cases}$$  \hspace{1cm} (5.2)

where $U$ and $U'$ are uniformly distributed random number sequences between 0 and 1, $\lambda$ is the arrival rate of the exponential distribution and the parameters $\alpha$, $\gamma$ and $\beta$ are the, lower limit, mode, and upper limit of the triangular distribution.

The signal mode is designed for debugging purposes. In this mode, the start event generates a single token at a time which is send to the associated BPMN elements. A subsequent token is generated when the previous token reached the process’ end event. From this follows that the signal mode can be used to analyse BP without concurrent tasks. In case of operating the simulation in the stochastic mode, the tokens are randomly generated with exponentially distributed time delay. An end event generates a log file that contains the picked up token data and destroys the incoming tokens. This file provides all data for subsequent risk figure computation (see section 5.4.4).
5.2. Basic Meta-Model

5.2.1.2 Activity

- java.lang.Object
  - com.xj.anylogic.Func
    * bpram.basics.ActiveObjectExtended
    * bpram.Activity

There are numerous kinds of BP actions which must be representable by the activity element. Thus, its generic AnyLogic™ representation is comprehensive (Figure 5.3). The Activity class is defined as a subclass of AnyLogics Active object class and consists of the following classes out of the AnyLogic Enterprise Library: queue class, selectOutput class, delay class, split class, combine class, matchQ class, exit class and the enter class.

The object is implemented in two modes: A "simple delay" mode pretends an activity to hold up a while (delay time) before executed successfully (Figure 5.4). In the simple delay mode, the arriving token is passed to a token queue before it enters the delay object. This queue prevents, that the tokens are not omitted if they arrive while the delay object is occupied. This mechanism is able to represent the situation, where a BP is executed several times in parallel, which may cause queueing problems.

![Diagram of Generic activity object in AnyLogic™](image)

**Figure 5.3:** Generic activity object in AnyLogic™
The more sophisticated "availability check" mode can only be performed successfully if all associated physical objects are available. Figure 5.5 outlines the interplay among activities, physical objects and the coordinating VTR component (see section 5.2.3.4).

The availability check implementation needs the design of "sub-token" and "drone" objects (Figure 5.5). If a token enters an activity object and the "simple delay" flag is set to false, it will be duplicated at the split object. The original token is then placed in a match queue while the duplicate (a sub-token in fact) is forwarded to the VTR in order to check the availability of all associated physical objects. The original token remains in the match queue until its sub-token is routed back from the VTR packed with all information needed. This information is appended to the token which is released from the activity and forwarded to the next step of the remaining process.

A printing job, for instance, can be modelled as an activity controlled availability check. Designated specific physical objects (printer, network, transmitting PC and its operator) are involved. In case the availability check mode is active, the objects’ availabilities are queried. As a result, the sub-algorithm returns information about the activity’s success. The specification of (recovery) actions succeeding an activity failure is laid down by the process model.
5.2. Basic Meta-Model

5.2.1.3 Gateway

Gateways are used to control the token flow through the process. Within this meta-model, three types of gateways have been realized: The XOR_Split (exclusive "OR"), the AND_Split and the AND_Combine gateway.

**XOR_Split** The XOR_Split is a branching element and serves as bifurcation for the sequence flow within a process. The XOR_Split has one flow input and two possible outputs (true / false). Each passing token is either guided to the first (true) or to the second (false) output. The XOR_Split class contains a selectOutput object from the *Enterprise Library*. The XOR_Split can be used in three different modes:

- **Global Availability Mode:** This mode is designed to make the XOR decision (true / false) depending on the success of the last activity in the process chain. Thus, it is checked, whether all the required physical objects of the last activity are available or not. If so, the corresponding token is guided to the output "true", otherwise, the token is released through the output "false".

- **Loop Count Mode:** To prevent the tokens to be caught in an infinite process loop,
the tokens monitor permanently the whole path passed through a process. In the loop count mode it is counted, how many times the token has passed this gateway and if a defined threshold is exceeded, the token is automatically passed to the output "true".

- **Probability Mode**: In this mode, the selected output depends on a defined probability $p \in \{0 \ldots 1\}$. For each passing token, a random number $x = \text{uniform}(1)$ is generated. If $x < p$ the output true is selected, otherwise the token is released through the output "false".

**AND_Split** The AND_Split gateway serves to split the sequence flow into two parallel flows. This class consists of the a split object of the *Enterprise Library*, which releases two identical (except the ID) copies of an incoming token. In order to link the two clones together, their IDs are made the same after the cloning. This enables a proper merging of the subdivided task (see AND_Combine). However, this splitting gateway has its limitations as e.g. an asymmetric sub-splitting is not supported.

**AND_Combine** The AND_Combine gateway is designed to recombine two split sequence flows. Therewith it combines all tokens with the same ID by extracting the collected data from the first token and attaching it to the second one. The second token is then released, while the first token is eliminated.

![Diagram](image_url)

**Figure 5.6**: Inner structure of the AND_Combine element as displayed by the AnyLogic GUI

The AND_Combine class consists of two *Enterprise Library* objects, a matchQ and a combine element. The matchQ consists of two separate token-queues. A token rests
in the queue, as long as in there is no token with the same ID in the opposite queue. As soon as the related token arrives, both tokens are released to the combine element.

5.2.1.4 End Event

The class *End Event* is always used at the end of a process chain of a model. One process can have several end events, but must have at least one.

The end event class contains a sink object from the *Enterprise Library* as main element. This element serves as a token eliminator. Before the tokens are eliminated, their data are stored in an output table for further analysis purposes.

5.2.2 Physical Components

To represent the "real world", physical components are considered within this meta-model. Obviously there are infinite types of physically existing and involved entities with numerous attributes and properties. However, for this meta-model, a generic physical object class was designed which serves as a representative for any imaginable involved entity.

5.2.2.1 Physical Object

Physical objects represent all kind of physically existing entities which are involved in the modeled BP, i.e. they may represent persons, e.g. operators, machines, buildings etc., which are needed for a successful performance of a given BP. They are classified by four variables:

- *Mean time to failure (MTTF)*: This indicates the average time a physical object is operating. It is used to generate the random variables TTF (time to failure) for the simulation of the BP according to Eq. 5.3.

- *Mean time to repair (MTTR)*: This indicates the average time a physical object remains a non-operating state. It is used to generate the random variables TTR (time to repair) for the simulation of the BP according to Eq. 5.4.

- *Pool size (PSZ)*: The pool size indicates the inner redundancy of a physical object. If for instance the physical object is a shop teller with the pool size 3, this means, that there are 3 tellers working at the same place.
• Replication \(N_{PO}\): In contrast to the pool size, the replication indicates the number of different physical objects of the same kind. Considering the same example as before including a replication factor of 2 means, that there are two independent groups of tree tellers each.

It is assumed, that each physical object fails after a certain time \(TTF\) and has to be replaced or repaired which causes a time period \(TTR\) where the object is not available. In fact, each physical object has two main states: an operating state (available) and a non-operating state (unavailable).

![Simplified statechart of a physical object](image)

**Figure 5.7: Simplified statechart of a physical object**

The state transitions takes place after the random time \(TTF\) (Time To Failure) and \(TTR\) (Time To Repair). They are exponentially distributed and are generated by a random number generator (Eq. 5.3 and 5.4),

\[
TTF = -MTTF \cdot \ln(U) \tag{5.3}
\]

\[
TTR = -MTTR \cdot \ln(U') \tag{5.4}
\]

where \(U\) and \(U'\) are uniformly distributed random number sequences between 0 and 1. Over an infinite period of time, the net probability \(p_F\) of finding a physical object in the non-operating state at any time is then given by:

\[
p_F = \left( \frac{MTTR}{MTTR + MTTF} \right)^{N_{PO}} \tag{5.5}
\]

The physical objects "communicate" with the process elements through the virtual components (see section 5.2.3). Thereto they are equipped with a code which handles
incoming availability requests. If an availability request (drone, see section 5.2.3.3) reaches a physical object while its operating state, the request is held for a certain time. This represents the execution of the corresponding activity. During this execution time, other incoming requests have to queue. If the physical object is in the non-operating state, the requests are sent back immediately.

5.2.2.2 Adjacency Matrix

The linkage between the process components and the physical components, i.e. the information, which activity requires, which physical object, is stored in an adjacency matrix:

<table>
<thead>
<tr>
<th></th>
<th>PO&lt;sub&gt;1&lt;/sub&gt;</th>
<th>PO&lt;sub&gt;2&lt;/sub&gt;</th>
<th>⋯</th>
<th>PO&lt;sub&gt;j&lt;/sub&gt;</th>
<th>⋯</th>
<th>PO&lt;sub&gt;k&lt;/sub&gt;&lt;sub&gt;−&lt;/sub&gt;1</th>
<th>PO&lt;sub&gt;k&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;sub&gt;1&lt;/sub&gt;</td>
<td>1</td>
<td>0</td>
<td>⋯</td>
<td>m&lt;sub&gt;1,j&lt;/sub&gt;</td>
<td>⋯</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A&lt;sub&gt;2&lt;/sub&gt;</td>
<td>0</td>
<td>0</td>
<td>⋯</td>
<td>m&lt;sub&gt;2,j&lt;/sub&gt;</td>
<td>⋯</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋯</td>
<td>⋮</td>
<td>⋯</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>A&lt;sub&gt;i&lt;/sub&gt;</td>
<td>m&lt;sub&gt;i,1&lt;/sub&gt;</td>
<td>m&lt;sub&gt;i,2&lt;/sub&gt;</td>
<td>⋯</td>
<td>m&lt;sub&gt;i,j&lt;/sub&gt;</td>
<td>⋯</td>
<td>m&lt;sub&gt;i,k&lt;/sub&gt;&lt;sub&gt;−&lt;/sub&gt;1</td>
<td>m&lt;sub&gt;i,k&lt;/sub&gt;</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋯</td>
<td>⋮</td>
<td>⋯</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>A&lt;sub&gt;n&lt;/sub&gt;&lt;sub&gt;−&lt;/sub&gt;1</td>
<td>1</td>
<td>1</td>
<td>⋯</td>
<td>m&lt;sub&gt;n&lt;sub&gt;−&lt;/sub&gt;1,j&lt;/sub&gt;</td>
<td>⋯</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A&lt;sub&gt;n&lt;/sub&gt;</td>
<td>0</td>
<td>0</td>
<td>⋯</td>
<td>m&lt;sub&gt;n,j&lt;/sub&gt;</td>
<td>⋯</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The adjacency matrix is an \( n \times k \) matrix, where \( n \) is the number of activities in the model and \( k \) represents the amount of involved physical objects. The matrix element \( m_{i,j} = 1 \) if the activity \( A_i \) requires the physical object \( PO_j \), else \( m_{i,j} = 0 \).

5.2.3 Virtual Components

5.2.3.1 Token

A token is a marker which indicates the actual progress state of a BP following the activities along the sequence flow. It can be interpreted as an abstraction of a task. It is a trigger object which also collects progress data passing activities ("simulation progress history data"). The token is a subclass of the "entity class" defined in the AnyLogic "Enterprise Library Reference Guide". It is a timed object, i.e., it ages while it
passes the BP. The tokens are released by start events. For more details, it is referred to the appendix.

A token $i$ may inclose $j$ subtokens and each subtoken itself may inclose $k$ drones. Within a simulation, the parameters $i$, $j$ and $k$ have the following meaning:

- $i$: The counter for the number of times a process is executed
- $j$: The token $i$ is located at its $j$-th activity. $j$ may be greater than the total number of activities if there are feedback loops in the process.
- $k$: The $j$-th activity needs $k$ physical objects to perform correctly.

### 5.2.3.2 Subtoken

The Subtoken class is designed as a first link between the process components and the physical objects. It investigates, whether all necessary physical objects of a given activity are available or not. The primary goal of each subtoken is to collect various types of data during the execution of a BP, therefore it contains several data structures to save these information:

- **Source-Activity Vector**: Every subtoken records its origin in the form of a source-activity vector before it leaves an activity. This information is needed by the VTR (section 5.2.3.4) to route back the subtoken to its actual activity after the performed availability check. When the subtoken returns to its origin activity, it is added to the waiting token, including all the stored information.

- **Target-Destination Vector**: Before the subtoken leaves an activity, it has to get the list of physical objects which have to be checked. This list is extracted from the adjacency matrix (see section 5.2.2.2) and is stored in the target-destination vector of the subtoken.

- **Drone Objects**: The subtoken itself is not passed to the physical objects, but only to the VTR. From there, it sends so called drones to the physical objects in order to analyze their state. The returning drones containing all the information about the checked physical object are completely stored in the subtoken.
5.2.3.3 Drone

The drones are generated by the VTR. For each subtoken, there are generated as many drones as physical objects are listed in its target-destination vector. The drones serve to request the status of physical objects which they save in a Boolean destination-availability parameter.

5.2.3.4 Virtual Token Router (VTR)

The VTR object is a message passing system. This object manages the forwarding of the requests (i.e., subtokens) from linked activities to physical objects. If a subtoken \( i \) arrives with the objective to check the state of specified physical objects, the VTR ensures that the subtoken receives all its information needed. It is evident that the VTR approach is close to a real network router operation.

Sequential (synchronous) and concurrent (asynchronous) checks are two methods to monitor state machines. The demand which mode has to be performed (sequential by default) is defined by an activity object. A synchronous check of physical objects is realised by generating sub-sub tokens ("drones"), whereas an available physical object can be occupied by a drone (Figure 5.8).

![Figure 5.8: Synchronous availability check of physical objects](image)

If a sub-token arrives at the VTR from the activity X, a drone will be generated and sent to the first associated physical object (e.g., a PC), while a sub-token is placed in a VTR match-queue. When the first drone comes back carrying all the first physical object's state-information the drone is merged with the sub-token. Further physical objects are checked by repeating this procedure. If a physical object is available at the drone's arrival, it will be temporarily occupied and sent back to the VTR then. If another activity's drone arrives, it is queued up as long as this physical object is occupied.
A lot of activities in a BP need *asynchronous checks*, i.e., the scan of physical objects’ availability is carried out in parallel. For this, the VTR simultaneously generates a drone for each physical object to be checked. These drones are sent to their destinations in parallel while the corresponding sub-token stays in a match-queue until all drones are returned to the VTR (Figure 5.9).

From the design point of view, the asynchronous feature’s implementation is found to be difficult. The proposed solution generates a separate match-queue for each sub-token. If a drone comes back to the router, it will be forwarded to the match-queue of its corresponding sub-token. If the number of drones in this match-queue is equal to the number of physical objects to be checked by the sub-token, the drones will copy all their information to the sub-token, which leaves immediately. The match-queue and all the drones are annihilated afterwards.

### 5.3 Risk Analysis

#### 5.3.1 Risk Estimation

IS and BP show many perspectives as various participants and actors are involved. The author proposes to choose and focus on a process owner’s view. In BP’s point of view, each divergence from a target value (i.e., a deviation from BP’s straightforward execution) is considered as an undesired event causing *consequences* as missed profit, unexpected expense, idle times and performance problems. As mentioned above, a token accumulates all information about regularities and irregularities during the process execution, e.g., within the BP shown in Figure 5.10.

Each simulation run rests on the number of initial tokens $n$ specified by the analyst. Each token $i = 1, 2, \ldots, n$ arriving at a BP’s end state releases a vector

![Figure 5.9: Asynchronous availability check of physical objects](image-url)
5.3. Risk Analysis

$\varepsilon_1 = \{S_{\text{end}}, n_{A,i}, C_{c,i}, A^{(1)}, A^{(2)}, \ldots, A^{(j)}\}$. $\quad (5.6)$

Thus, $v_i$ characterises a scenario. The vector’s parameters are introduced below and are finally used for risk estimation purposes.

Boolean $S_{\text{end}}$ indicates whether a BP succeeded or failed, i.e.,

$$S_{\text{end}} = \begin{cases} 0 & \text{success\_END} \\ 1 & \text{failure\_END} \end{cases}. \quad (5.7)$$

The $\varepsilon_i$-parameter $n_{A,i}$ gives the number of activities involved in the scenario $i$. Each activity of the BP is rated by an irregularity value $Ir$ (namely 1 for regular activities and up to a predefined value (e.g., 10) for irregular activities). In respect to Figure 5.10, the regular BP follows $\text{START} \rightarrow A1 \rightarrow A2 \rightarrow \text{successful\_END}$ and thus activities $A1$ and $A2$ are rated with $Ir = 1$. If $A1$ fails, the irregular activity $A3$ is performed and rated, e.g., $r_{A3} = 2$. If $A3$ fails as well, either activity $A4$ (e.g., $Ir_{A4} = 10$) is performed or the process stops with $\text{failure\_END}$.

The accumulated consequences $C_i$ of a scenario $i$ are computed by Equation 5.8:

$$C_i = C_{c,i} + C_{L,i} + C_{R,i}. \quad (5.8)$$

The rated costs $C_{c,i}$ are accumulated by Equation 5.9
\[ C_{C,i} = \sum_{j=1}^{m} Ir_j, \]  

where \( j = 1, \ldots, m \) indicates an activity specifically involved in a BP’s scenario that is rated by \( Ir \); \( C_C \) = estimated process costs and losses; \( C_L \) = missed profit; and \( C_R \) = not yet specified costs.

Equations 5.7 and 5.10 give the missed profit \( C_{L,j} \), i.e.,

\[ C_{L,j} = S_{end} \cdot Im_j, \]  

where \( Im \) = specified (i.e., expected) benefit.

If, for instance, an enterprise does not place a contract with a client due to a bad BP performance this is rated within the simulation as an undesired event resulting in not having made the specified benefit \( Im \).

Following Equation 5.8, the overall risk \( R \) is

\[ R = R_C + R_L + R_R, \]  

which is the sum of the expected values of the individual consequences \( C_C, C_L, \) and \( C_R \).

Thus Equation 5.11 becomes

\[ R = \frac{1}{n} \sum_{i=1}^{n} C_{C,i} + \frac{1}{n} \sum_{i=1}^{n} C_{L,i} + \frac{1}{n} \sum_{i=1}^{n} C_{R,i}, \]  

that is

\[ R = \frac{1}{n} \sum_{i=1}^{n} C_i. \]  

A F\( \text{C-Diagram} \) representation of the results also needs the estimation of the probabilities

\[ P(C_i = k) = \frac{n(C_i = k)}{n}. \]  

### 5.4 Case Study "Ticket Reseller"

The simulation of a major ticket reseller’s IS exemplifies the feasibility of the presented risk analysis methodology. The corresponding ticket-reseller offers tickets for various
events. Customers can order and buy the tickets in a shop in the city (ticket-shop), an online shop or they can order them by phone (call-center). The interlinked processes are an online process, a call-centre process, and a ticket-shop process.

\[\text{Figure 5.11: The tree major processes of the "Ticket Reseller"}\]

5.4.1 Setting up the BP Model

In the model, these tree processes are inclosed in separate, interlinked sub-processes (see Figure 5.11), each of it with its own start event, but all of them with the same end events (success or failure). Within this example model, all three processes are highly simplified in order highlight the details of the modeling approach. The processes are composed of a total of 16 activities and 19 XOR gateways:

5.4.1.1 Online Process

The simplified online process is represented by its tree main activities: 1.) connecting to the site, 2.) searching for a desired ticket and 3.) booking the ticket (see Figure 5.12, left side). If all tree activities are successful, the tickets are booked within tree steps. However, each of the mentioned activity may also fail. For this case, a generic debugging sub-process is modeled (see Figure 5.12, right side).
5.4.1.2 Call-Center Process

Although a lot of people prefer to order and book their tickets from their home, not all of them have access to the Internet. Therefore, most shops offer an "order by phone" service. The people and the necessary equipment for this service may be either situated at the same place as a shop, or at a different place, e.g. a call-center. In the model, this two situations are differentiated by the association of physical objects either as a pool (first situation) or as several replicated (second situation). The modeled call-center process is displayed on the left side of Figure 5.13. It contains only the two most important activities: 1.) call shop, 2.) book ticket.

5.4.1.3 Ticket Shop Process

When people go to the shop to buy their tickets, the booking process is enlarged by two new requirements: 1.) The tickets have to be payed immediately and 2.) the tickets have to be printed on-site (see the ticket-shop process on the right side of Figure 5.13.)

5.4.1.4 Process Summary

For each activity of the above described process, a simulation mode (availability check / simple delay) has to be defined, and for those activities performing an availability check, the corresponding physical objects have to be assigned. Table 5.2 gives an overview of all the 16 modeled activities including their modes and irregularity parameters $I_r$. The
assigned objects are discussed in section 5.4.3 and the corresponding linkage information is stored in the Adjacency Matrix (see Table 5.4). The estimation and quantification of the irregularity parameters is done by a simple expert judgement. Accordingly, these values are not indisputable within the presented case study. The determining of the irregularity parameters should be done carefully, as they have a strong influence on the risk analysis performed with this approach. However, this values can be adjusted if it turns out after the simulation, that they where chosen inadequately.

5.4.2 Defining the Physical World

Shared media are a database, a database server, an application server, an ISP network (internet service provider), and the Internet. Other "objects" involved are the shops' and call-centres' computers, printers, phones, etc., and staff, e.g., tellers and helpline consultants. A detailed list of all modeled physical objects is given in Table 5.3.

Arbitrary estimators of reliability parameters (i.e. MTTF and MTTR) are used in this case study as well as the average customers’ rate for each sub-process. (These estimators and distribution functions are not verified in this paper. For simplification purposes exponential distributions are used. However, the model as well as AnyLogic™ copes with any kind of distributions.)
Table 5.2: Activities modeled within the Case Study (avck = availability check mode)

<table>
<thead>
<tr>
<th>ID</th>
<th>Ir</th>
<th>Mode</th>
<th>Process</th>
<th>Sub-Activity</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1</td>
<td>avck</td>
<td>ticket-shop</td>
<td>book ticket</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>1</td>
<td>avck</td>
<td>ticket-shop</td>
<td>print ticket</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>1</td>
<td>avck</td>
<td>ticket-shop</td>
<td>pay ticket</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>3</td>
<td>simpl delay</td>
<td>ticket-shop</td>
<td>try again</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>4</td>
<td>simpl delay</td>
<td>ticket-shop</td>
<td>repair printer</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>2</td>
<td>simpl delay</td>
<td>ticket-shop</td>
<td>change payment</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>1</td>
<td>avck</td>
<td>online</td>
<td>connect to site</td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>1</td>
<td>avck</td>
<td>online</td>
<td>search ticket</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>2</td>
<td>simpl delay</td>
<td>online</td>
<td>debugging</td>
<td>think</td>
</tr>
<tr>
<td>A10</td>
<td>2</td>
<td>simpl delay</td>
<td>online</td>
<td>debugging</td>
<td>turn off PF</td>
</tr>
<tr>
<td>A11</td>
<td>3</td>
<td>simpl delay</td>
<td>online</td>
<td>debugging</td>
<td>try again</td>
</tr>
<tr>
<td>A12</td>
<td>3</td>
<td>avck</td>
<td>online</td>
<td>debugging</td>
<td>call helpline</td>
</tr>
<tr>
<td>A13</td>
<td>1</td>
<td>avck</td>
<td>online</td>
<td>book ticket</td>
<td></td>
</tr>
<tr>
<td>A14</td>
<td>1</td>
<td>avck</td>
<td>call-center</td>
<td>call shop</td>
<td></td>
</tr>
<tr>
<td>A15</td>
<td>1</td>
<td>avck</td>
<td>call-center</td>
<td>book ticket</td>
<td></td>
</tr>
<tr>
<td>A16</td>
<td>3</td>
<td>simpl delay</td>
<td>call-center</td>
<td>try again</td>
<td></td>
</tr>
</tbody>
</table>

5.4.3 Adjacency Matrix

Most activities of the BP depend on physical objects of the real world. For instance to be able to buy a ticket in the ticket-shop, a lot of physical requirements have to be fulfilled: Shop tellers have to be in place, their PC has to work correctly, the networks (shops Intranet, ISP-network and the internet) need to operate as the shop tellers need to access the ticket-database on the database-server, which is located in a secure server-room apart from the shop. In addition, the printer has to operate correctly, as the tickets are printed directly on site. All these information are stored in the adjacency matrix (see Table 5.4).

5.4.4 Simulation and Results

A simulation generates a plethora of log file information (see Figure 5.14). In the header of this file, all the logged activities $A_i$ are listed and denominated. After a delimiterator, the simulation data, i.e. the scenario vectors $v_i$, is listed. The first entry $(0,3,3,1,2,3)$ in the excerpt of a log-file (Figure 5.14) denotes for instance that the first customer
Table 5.3: Properties of the physical objects (see section 5.2.2.1) within the Case Study: MTTF and MTTR are declared in "minutes", $N_{PO} =$ replication, $P_{SZ} =$ pool size, $p_F =$ failure probability (Eq. 5.5).

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Physical Obj.</th>
<th>MTTF</th>
<th>MTTR</th>
<th>$N_{PO}$</th>
<th>$P_{SZ}$</th>
<th>$p_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PO_1$</td>
<td>human</td>
<td>shop tellers</td>
<td>1000</td>
<td>200</td>
<td>1</td>
<td>3</td>
<td>0.46%</td>
</tr>
<tr>
<td>$PO_2$</td>
<td>call tellers</td>
<td>500</td>
<td>200</td>
<td>1</td>
<td>4</td>
<td>0.67%</td>
<td></td>
</tr>
<tr>
<td>$PO_3$</td>
<td>consultants</td>
<td>300</td>
<td>300</td>
<td>1</td>
<td>3</td>
<td>12.50%</td>
<td></td>
</tr>
<tr>
<td>$PO_4$</td>
<td>network</td>
<td>Internet</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
</tr>
<tr>
<td>$PO_5$</td>
<td>intranet shop</td>
<td>50000</td>
<td>300</td>
<td>1</td>
<td>1</td>
<td>0.60%</td>
<td></td>
</tr>
<tr>
<td>$PO_6$</td>
<td>ISP-network</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
<td></td>
</tr>
<tr>
<td>$PO_7$</td>
<td>hardware</td>
<td>app-server</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
</tr>
<tr>
<td>$PO_8$</td>
<td>DB-server</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
<td></td>
</tr>
<tr>
<td>$PO_9$</td>
<td>telephones</td>
<td>20000</td>
<td>100</td>
<td>5</td>
<td>1</td>
<td>0.50%</td>
<td></td>
</tr>
<tr>
<td>$PO_{10}$</td>
<td>PCs</td>
<td>3000</td>
<td>100</td>
<td>5</td>
<td>1</td>
<td>3.23%</td>
<td></td>
</tr>
<tr>
<td>$PO_{11}$</td>
<td>printers</td>
<td>3000</td>
<td>100</td>
<td>3</td>
<td>1</td>
<td>3.23%</td>
<td></td>
</tr>
<tr>
<td>$PO_{12}$</td>
<td>software</td>
<td>database</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
</tr>
<tr>
<td>$PO_{13}$</td>
<td>app-SSL</td>
<td>10000</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0.99%</td>
<td></td>
</tr>
</tbody>
</table>

succeeded to buy the ticket ($S_{end} = 0$). To get the ticket, three activities were performed in this scenario ($n_{A,1} = 3$). As the total costs of the scenario ($C_{C,1} = 3$) have the same value as $n_{A,1}$, there was no deviation from the optimal path through the process. This can be verified by checking the irregularity parameters $I_{Ir}$ of the performed activities ($A_1$, $A_2$, $A_3$) within this first scenario which all have a value of 1 and belong to the "ticket-shop process" (see Table 5.2).

5.4.4.1 The Simulation Experiment

Risk analysis is interested in failed BP whereas established BP modelling concentrates on successfully implemented BP: A sole simulation meets demands of usually separated business fields. This feature may increase the acceptance of risk analysis in enterprises management in general.

For this case study, a simulation experiment of 49'385 customer requests (tokens) was performed. The simulation consumed a time of approximately 20 minutes and was performed on an Intel Pentium Core 2 Duo (2.8 GHz, 1 GB RAM) computer running under Windows XP. Within this time, almost one million entities where generated and
Table 5.4: Adjacency Matrix of the Case Study: \( A_i = \text{activities}, \ PO_j = \text{physical objects} \)

<table>
<thead>
<tr>
<th></th>
<th>PO_1</th>
<th>PO_2</th>
<th>PO_3</th>
<th>PO_4</th>
<th>PO_5</th>
<th>PO_6</th>
<th>PO_7</th>
<th>PO_8</th>
<th>PO_9</th>
<th>PO_10</th>
<th>PO_11</th>
<th>PO_12</th>
<th>PO_13</th>
<th>PO_14</th>
<th>PO_15</th>
<th>PO_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A_9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A_14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_15</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A_16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

processed by the simulation algorithm (see Table 5.5). During this simulation, also a lot of dispensable data for debugging purposes where logged in a separate file (2-3 MB per second). If only the output of interest (see Figure 5.14) is logged, the time consumption can be reduced significantly.

Table 5.5: Summary of the simulation experiment

<table>
<thead>
<tr>
<th>Entity</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokens</td>
<td>49'385</td>
</tr>
<tr>
<td>Sub-Tokens</td>
<td>158'541</td>
</tr>
<tr>
<td>Drones</td>
<td>708'716</td>
</tr>
<tr>
<td>Total Entities</td>
<td>916'642</td>
</tr>
</tbody>
</table>
5.4. Case Study "Ticket Reseller"

Figure 5.14: Logfile entries (excerpt)

ID mapping <-> Activity Name <-> Ir
1 root.buyInCityshop.bookTicket, Ir=1
2 root.buyInCityshop.printTicket, Ir=1
3 root.buyInCityshop.payTicket, Ir=1
4 root.buyInCityshop.tryAgain, Ir=3
5 root.buyInCityshop.repairPrinter, Ir=4
6 root.buyInCityshop.changePayment, Ir=2
7 root.buyOnline.connectToSite, Ir=1
8 root.buyOnline.searchTicket, Ir=1
9 root.buyOnline.debugProblem.think, Ir=2
10 root.buyOnline.debugProblem.turnOffPc, Ir=2
11 root.buyOnline.debugProblem.tryAgain, Ir=3
12 root.buyOnline.debugProblem.callHelpline, Ir=3
13 root.buyOnline.bookTicket, Ir=1
14 root.buyByPhone.callShop, Ir=1
15 root.buyByPhone.bookTicket, Ir=1
16 root.buyByPhone.tryAgain, Ir=3

0, 3, 3, 1, 2, 3,
...
1, 10, 17, 7, 8, 13, 9, 11, 7, 9, 11, 7, 9,
1, 7, 13, 1, 4, 1, 4, 1, 4, 1,
...

5.4.4.2 Process Performance

A first brief analysis of the data already highlights the overall success rates of the modeled process (see Table 5.6). The failure rate of more than 20% is obviously not realistic. This roots in the arbitrary chosen parameters for the single failure rates of the physical object which are overestimated. Although, comparing the relative frequencies of, e.g., the different scenarios is of interest as this gives an insight in the process behavior and the highlights the main cost drivers.

<table>
<thead>
<tr>
<th>Table 5.6: Success rates of the case process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
<tr>
<td>Tokens</td>
</tr>
<tr>
<td>Fraction (rates)</td>
</tr>
</tbody>
</table>
The average emerging costs per customer (risk) is represented by $R_C$. In case of successful BPs, the computed average costs per token are 3.00, in case of failed processes the average costs are 7.73. The total expected costs is $R_C = 3.98$.

### 5.4.4.3 Scenario Analysis

To analyze the process data in detail, the log-files are used to compute the risk numbers (e.g. the expected costs $C_C$) presented in section 5.3 and allocated to those scenarios causing this costs. The resulting cumulative frequencies per cost are presented in Figure 5.15. For 96.32% of the successful scenarios $C_C \leq 3$ where on the other side, 38.59% of the failed scenarios have a $C_C > 12$. This is interesting from the perspective of risk analysis as it indicates high frequencies by simultaneous high consequences.

![Cumulative Probability](image)

**Figure 5.15: Cumulative "cost"-probability**

The cause for this weighty fraction of high consequences (costs) for failed processes can be analyzed in detail by determining, which scenarios and activities are responsible for this. For this purpose, the histogram of the emerged process costs $C_C$ (Figure 5.16) can serve as a helpful plot. It shows about 28.76% of the failed scenarios result in a $C_C = 13$. This corresponds to 2992 Tokens or 6.06% of all simulated scenarios.

In order to identify the responsible process scenarios causing this high costs of $C_C = 13$, the data has to be analyzed in more detail. Therefore, all vectors $v_i$ of the log-file containing a $C_C$ of 13 are grouped according their scenarios. The result of this classification is summarized in Table 5.7. Obviously, more than 90% of the specified
scenarios are composed of a single loop in the process between the two activities $A_1$ and $A_4$.

**Table 5.7: Fractions of the scenarios resulting in $C_C = 13$.**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Fraction</th>
<th>Sequence of performed activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$, (failure)</td>
<td>93.45%</td>
<td>$A_1, A_4, A_1, A_4, A_1, A_4, A_1$</td>
</tr>
<tr>
<td>$s_2$, (failure)</td>
<td>3.04%</td>
<td>$A_{14}, A_{16}, A_{14}, A_{16}, A_{14}, A_{16}, A_{14}$</td>
</tr>
<tr>
<td>$s_3$, (failure)</td>
<td>2.11%</td>
<td>$A_7, A_8, A_{13}, A_9, A_{11}, A_7, A_8, A_{13}, A_9$</td>
</tr>
<tr>
<td>$s_4$, (failure)</td>
<td>0.03%</td>
<td>$A_{14}, A_{15}, A_1, A_4, A_1, A_2, A_5, A_2$</td>
</tr>
<tr>
<td>$s_5$, (success)</td>
<td>1.37%</td>
<td>$A_{14}, A_{15}, A_1, A_4, A_1, A_4, A_1, A_2, A_3$</td>
</tr>
</tbody>
</table>

The analysis of the frequencies of activity-execution and their corresponding irregularity parameters as consequence estimators results in a type of frequency-consequence diagram (Figure 5.17). Consistent with the scenario analysis, the activity $A_4$ stands out as it shows a high execution-frequency and at the same time a relatively high irregularity.

Subsequent to the risk analysis, it is the task of IS and BP optimisation to decrease the impact of identified major scenarios, e.g., by BP re-engineering, or increasing the availability of involved physical objects. The impact of an optimisation measurement is assessed by minimising the total risk $R$ (Equation 5.12).
Figure 5.17: Frequencies of activity-execution against their irregularity.
Chapter 6

Conclusions

Engineered systems have witnessed greater and higher integration and escalating complexity whereas risk and reliability aspects have become increasingly important. Established risk analytical methods - mainly allowing a static approach of decomposition and describing the system’s behavior by "the sum of the behavior of its parts" - turned out to be insufficient to model and analyze such systems.

This thesis contributes to the development of risk analysis techniques based on a systems approach. Thereby, extensive literature researches have been made in order to identify the state of the art methodologies in system modeling in general. Agent-based modeling (ABM) is often put forward as a promising approach for modeling complex systems. Surprisingly, ABM is reluctantly used in the area of risk and reliability as a literature review clearly indicated.

As most of the scanned methods are atypical in their application to risk analysis, several case studies have been made to demonstrate applicability. Finally, an innovative methodology including a software library for its efficient application has been developed and tested. The implemented approach includes a business process modeling (BPM) interface for system modeling and is merged with ABM technology for computer simulation.

Based on the author’s experience, BPM provides a manageable procedure for analyzing complex and networked systems. However, there are numerous BPM techniques in use and all of them have specific properties. In order to find a suitable BPM notation for risk analysis, a pre-study was carried out. During this study, a simple analysis approach was developed and is presented within this thesis. This approach focuses on the information system (IS) – rather than on the computer network behavior. As a consequence, critical (business) functions are identified in place of, e.g., critical hardware and software. Then,
high risked ranked functions can be analysed in detail, e.g., by using established risk analysis techniques such as fault tree analysis (FTA). This approach is thus considered as a risk ranking and prioritization tool. It shows practical outcomes within the modeling assumptions and simplifications.

As a next step, three case studies have been carried out to check the suitability and feasibility of ABM in risk analysis. These studies clearly indicate that ABM is a very powerful approach.

CaseStudy 1: The introduced virus propagation model, which includes complex multi-agent interactions, demonstrates how such models can be used for testing countermeasures against emerging risks.

CaseStudy 2: The feasibility and correctness of ABM results are illustrated by solving a "classical" reliability (availability) problem including maintenance (repair) work and finally by comparing a Markov model with ABM. Both models use Monte Carlo simulation techniques for quantification. The ABM approach turned out to be applicable and to deliver correct results, i.e., comparable with the other modeling approach for same conditions. Moreover, the ABM model was successfully extended to include various maintenance strategies. It also demonstrates its capability to cope with dynamic system behavior and to detect "hidden" system properties. First efforts have been made to use ABM for full scope risk analysis purposes including a cost-benefit analysis.

CaseStudy 3: To investigate the capabilities of ABM to model cascading failures in complex networks, a generic model which considers two stress induced component outage types has been successfully developed.

Finally, the gained know-how was used to develop a novel risk analysis approach, which is based on the agent-based simulation of business processes (BP), performed on physical IS. Thereby, a software library has been compiled which serves as a simulation tool for risk analysis. Core elements of this library are 1) BP objects, which enables a modeler to set up the BPM by drag and drop, 2) generic physical objects to represent, e.g., hardware and its failure mechanism and 3) an extended simulation engine (Virtual Token Router) which enables the proper devolution of the computer simulation. It turned out, that this approach shows promising aspects:

- The generation of simple BPM usually is trouble-free. The quantification of BPM-Risk analysis parameters is supported by simple categorizing lists or pure expert judgement but estimation remains still challenging.

- The final methodology allows an analyst to apply a top down approach in modeling. Consequently, the models’ granularity is increaseable step by step accordant
to demands, knowledge, and resources.

- For an experienced user, it takes about one day to realise a simulation of a medium sized IS (when BP and reliability data are known). This is a very short time compared to common simulation tools in risk analysis.

- The transformation of the semi-formal BPMN framework to the formal simulation model is considered as a major innovation. The methodology provides the automated generation of scenarios and many risk relevant data sets. The amount of data allows the application of sophisticated analysis approaches from, e.g., multivariate statistics, data mining, and network theory.

However, it is indisputable, that some challenges remain unsolved and that maybe this approach is not able to solve all the problems which it was intended to address:

- Bad performance is a well known problem in computer based simulation. The performance of the simulation allows to handle medium sized systems. However, systematical parameter variation and optimization is out of scope. This kind of operations would imply that the simulation has to run several thousand times, each time with different parameter settings. Assuming the 20 minutes per simulation, this would last practically to long.

- Too many (unknown) parameters: In order to get a valuable output of the model, the numerous input parameters (e.g., failure rates, etc.) have to be determined properly. Especially the estimation of the irregularity parameters turned out to be quite difficult as the parameter has to be cross-comparable for all kind of activities.

- Modeling from the bottom up is powerful to reproduce complex system behavior, but it has also its disadvantages: There have to be made a lot of assumptions, e.g., about external processes (e.g., customers’ or service providers’ decision sequences) which are modeled by separate agents interacting with the own IS.

- Too many cases: The compiled method is able to handle a good deal of IS and their corresponding processes. However, it is not able deal with all kind of structures without being adapted. This roots in a trade off between user-friendliness of the software and capability of the method. The software library has been implemented as an easy to use drag- and drop library plug in for the AnyLogic simulation toolkit. "Easy to use" means, that the user do not have to write program code in Java himself, but only model the system and define the system properties.
• The predictive power of a simulation-based model depends on the number of input parameters. The use of too many variables results in a lower predictive power of the simulation output. A modeler must be aware of this fact and try to reduce the amount of parameters, e.g., by using the same mean failure rates for the same type of machines.

• The verification of such complex simulation models is very important. Therefore a debugging mode (see signal mode of the Start Event section 5.2.1.1) has been integrated into the simulation environment in order to inspect the simulation step by step and to make sure it works correctly. However, it turned out that this kind of verification is very time-consuming. As a major future improvement, an automated verification system should be integrated into the simulation environment. Such a system should be able to monitor and verify the behavior of every agent automatically [142].

• Validation of these kinds of simulation models is very important but difficult as long as the behavior of the reference system is not well known. At the moment, the presented simulation environment includes no sophisticated validation support. It is recommended to validate the simulation model for each case study by applying established methods like face validation (expert validation), parameter calibration, sensitivity analysis and statistical validation [143]. However, in practice the missing of empirical data often reduces the possibilities.

• The representation of individuals as "physical objects" is not very realistic. It could be extended according to the standards in human reliability analysis.

In summary, the presented method for analyzing risk by ABM of BP is considered as a powerful and sophisticated tool compared to the time it consumes for analyzing a given system. It delivers a detailed insight in, e.g., an IS and helps to allocate potential risks and problems. Further more, it allows the modeler revealing unexpected emerging system properties, which is regarded as one of the major strengths of this approach. Thus, the feedback from business practitioners in the field of IS risk analysis has been very positive. They pretend to consider the user-friendliness combined with this level of sophistication as a precondition for a successful approach.
## List of Tables

2.1 Market share of the home computer operating systems in September 2007 (Ref: marketshare.hitslink.com) ........................................ 11

3.1 Rating of seven mayor BPM techniques ........................................ 28

3.2 Categories of probabilities of activity $i$ ........................................ 38

3.3 Summary of the Case Study results .............................................. 40

3.4 Summary of the Case Study results .............................................. 41

4.1 Investigated maintenance strategies. .............................................. 60

5.1 Adjacency Matrix: ($A_i =$ activities, $PO_j =$ physical objects) ........... 79

5.2 Activities modeled within the Case Study (avck = availability check mode) 88

5.3 Properties of the physical objects (see section 5.2.2.1) within the Case Study: MTTF and MTTR are declared in "$minutes". $N_{PO} =$ replication, $P_{SZ} =$ pool size, $p_f =$ failure probability (Eg. 5.5). ............................. 89

5.4 Adjacency Matrix of the Case Study: ($A_i =$ activities, $PO_j =$ physical objects) 90

5.5 Summary of the simulation experiment ...................................... 90

5.6 Success rates of the case process .............................................. 91

5.7 Fractions of the scenarios resulting in $C_C = 13$. .......................... 93

A.1 Library Objects ................................................................. 121

A.2 Special Objects (bound to this library) ...................................... 122

A.3 Field Summary ................................................................. 124

A.4 Method Summary ................................................................. 127
## List of Figures

1.1 Risk management system [2] .............................................. 3
1.2 Basic modeling framework; An agent-based model of abstract business processes and their underlying physical structure. .................. 8

2.1 Scenario frequency $f_{s,i}$, consequence probability $p_i$ and consequence frequency $f_{c,i}$ .................................................... 10
2.2 Classical risk analysis procedure ........................................... 13
2.3 Two classification types of scientific models ............................ 17
2.4 Classification of modeling approaches .................................... 19
2.5 Example of a system variable (performance) as an observable for consequence analysis ......................................................... 20
2.6 Classical risk analysis procedure ........................................... 21
2.7 The concept of deriving frequencies and consequences by "infinite" time simulation. The undesired events emerge during the simulation experiment 21

3.1 BPMN example ............................................................... 32
3.2 Extended BPMN. ............................................................. 33
3.3 Sample process containing one loop. ................................. 34
3.4 BPMN model of the "Software Security Vulnerability Management" process of a major swiss Internet service provider. For the activities $A_1 \ldots A_6$, the process is linear and therefore summarized by a single activity. 38
3.5 ETA of the failure event $F_4$ of Activity $A_4$. The resulting average consequence $C_{F,4} = 27$. ................................. 39
3.6 Correlation between the prioritization number $S_i$ and the risk number $R_{A,i}$. The coefficient of correlation is $r^2 = 0.6$ if the outlier of $A_2$ (black point) is neglected. ................................................................. 40

4.1 Agent properties ................................................................. 45

4.2 Agents state-chart ............................................................... 49

4.3 Modelled agents on a 400x400 m square field .......................... 50

4.4 Characteristic devolution of an epidemic. $p_{\text{mort}} = 0.1$, $\lambda_c = 5 \text{ week}^{-1}$, $\lambda_{\text{incub}} = 1 \text{ week}^{-1}$, $R_c = 25m$, $N = 500$, $\Delta t = 0.5 \text{ weeks}$, $m = 4$ .............................. 51

4.5 Distribution of modeled fatalities caused by an epidemic. $p_{\text{mort}} = 0.1$, $\lambda_c = 5 \text{ week}^{-1}$, $\lambda_{\text{incub}} = 1 \text{ week}^{-1}$, $R_c = 25m$, $N = 500$, $\Delta t = 0.5 \text{ week}$, and (A) $MR = 9972$, $m = 0$, (B) $MR = 3501$, $m = 2$, (C) $MR = 2712$, $m = 4$ ($MR = \text{Model Runs}$) .................................................. 52

4.6 Cumulative mortality per event for the tree scenarios $A$, $B$, and $C$ (left graph) and the influence of the mobility parameter $m$ on the risk (right graph) ................................................................. 53

4.7 Relative availability $\frac{p(N_O)}{p}$ of a single unit in a $n$-unit system, where $n = 50$ and $\rho = \frac{\lambda}{\mu} = 1$, i.e. $p = 0.5$ (see Eq. 4.5). .......................... 54

4.8 Two state model of the unit $i$ for Monte Carlo simulation ................. 55

4.9 Agent-based model of the units. ............................................. 57

4.10 Agent-based model of the operators. ......................................... 57

4.11 Sample of a model configuration: $n = 32$ state machines and $N_O = 5$ operators. ................................................................. 59

4.12 Agent-based model of the operators. ......................................... 59

4.13 Relative availability $\frac{p(N_O)}{p}$ of a single unit in a $n$-unit system maintained by different strategies ($S_0 \ldots S_4$), where $n = 50$ and $\frac{\mu}{\lambda} = 1$ (left graph) and $\frac{\mu}{\lambda} = 10$ (right graph). ................................................................. 61

4.14 Relative mean time to target ($\text{MTTT}/\text{R}_{[\text{v}]}$) against the number of operators $N_O$, ($\frac{\mu}{\lambda} = 1$, $n=50$, Strategy S2). .............................. 62
4.15 Cost-benefit analysis of the system \( n = 50, \frac{L}{\lambda} = 10, L = C = 1 \), applying the maintenance strategies S2 \( (N_O < 20) \) and S \( (N_O \geq 20) \), displaying the maintenance costs (dotted line), the expected loss (thin continuous line) and the sum of both (bold continuous line) left graph: Optimal number of operators (same model parameters) for different fractions of L/C right graph: .................................................. 63

4.16 Example of a generated random network (left side) and a scale-free network (right side) ................................................................. 66

4.17 Failure sequence on scale-free networks with \( MTTF = 1 \) year for three values of \( \langle T_{i,\text{crit}} \rangle = 0.4, 0.5, 0.6 \) (dotted, dashed and continuous line, respectively) .................................................. 67

4.18 Individual failure modes corresponding to Fig. 4.17, whereas the three larger slopes represent the failures due to reaching \( T_{i,\text{crit}} \) and the flat slopes the outages due to wearout ........................................... 67

4.19 The failure-rate \( \lambda(t) \) during a cascade in a scale-free network for \( \langle T_{i,\text{crit}} \rangle = 0.5 \) and different \( MTTF \), i.e. 1 week, 3 months, 6 months and 1 year (triangles, squares, diamonds and circles respectively) ......................... 68

5.1 Agent view of the BP representation of a generic IS .......................... 70

5.2 Components of the meta-model ..................................................... 71

5.3 Generic activity object in AnyLogic™ ......................................... 73

5.4 Simple delay mode .................................................................................. 74

5.5 Availability check .................................................................................... 75

5.6 Inner structure of the AND_Combine element as displayed by the AnyLogic GUI ................................................................. 76

5.7 Simplified statechart of a physical object ........................................... 78

5.8 Synchronous availability check of physical objects ................................. 81

5.9 Asynchronous availability check of physical objects ............................... 82

5.10 Simple business process ....................................................................... 83

5.11 The tree major processes of the "Ticket Reseller" ............................... 85

5.12 Ticket Reseller: Online process and debugging sub-processes ................. 86

5.13 Ticket Reseller: Call center and city shop sub-processes ....................... 87
5.14 Logfile entries (excerpt) ................................. 91
5.15 Cumulative "cost"-probability .............................. 92
5.16 Consequence estimation of failed BP execution ........... 93
5.17 Frequencies of activity-execution against their irregularity. 94

A.1 Basic principle of the presented meta-model .................. 120
A.2 Class diagramm of the main bpram objects activity and physical object . 124
A.3 Illustration of a very simple business process .................. 131
A.4 Class diagramm of the implemented bpram events ............ 132
A.5 Class diagramm of the implemented bpram gateways ........ 134
A.6 Class diagramm of the implemented bpram tokens and drone .... 137
A.7 Class diagramm of the implemented basic bpram objects ........ 138
List of Abbreviations

ABM  Agent-based Model
ABS  Agent-based Simulation
ALE  Annual Loss Exposure
ARIS Architektur integrierter Informationssysteme
BP   Business Process
BPD  Business Process Diagramm
BPM  Business Process Model
BPMN Business Process Modeling Notation
BPR  Business Process Reengineering
CAD  Computer-aided Design
EPC  Event-driven Process Chain
ERM  Entity Relationship Model
ETA  Event Tree Analysis
FC   Flow Charts
FHA  Fault Hazard Analysis
FMEA Failure Mode and Effects Analysis
FTA  Fault Tree Analysis
ICT  Information and Communication Technology
IS   Information System
IT   Information Technology
ITIL Information Technology Infrastructure Library
KTI/ICT Swiss Innovation Promotion Agency
LSA  Laboratory for Safety Analysis
MAS  Multi-agent System
PRA  Probabilistic Risk Analysis
UML United Modeling Language
Bibliography


[133] Sun, J.G., Yang, X.B. and Huang, D. Multi-agent Based Distributed Chemical Process Monitoring and Diagnosis. In First Int. Conf. on Machine Learning and Cybernetics, pages 851–856, Beijing, 2002. IEEE.


Appendix A

Library Reference Guide

A.1 General Principles

This appendix presents details of the business process modelling and simulation library for a sophisticated risk analysis of complex information systems (IS), which was developed within this thesis. The library is implemented as an Add-In for the simulation software AnyLogic and its GUI. A symbolic object library is compiled and a user connects the BPMN objects by drag and drop. The basic concept of this approach is to model and simulate the "real world" in order to observe its behaviour with focus on the frequency and consequence of emerging events. The modelling of an IS by means of this library encompasses three steps.

- Business process (BP) modelling using BPMN
- Physical object (PO) modelling
- Defining the interdependencies among BP and PO (\(\rightarrow\) Adjazenzmatrix)

In this modelling framework, network components (computers, phones,"staff", etc.) are referred to as physical objects. As shown in Figure A.1, the information exchange between the BP model and PO model is managed by a Virtual Token Router (VTR) (see section 3.11). If a PO is unavailable, the VTR makes this information available in the simulation. The business process simulation is realised by the implementation of an entity flow similar to the Petri-net approach. One task within a process is represented by one token. In order to perform an availability-check of the physical objects, the token generates sub-tokens which are sent to the VTR. There, the sub-tokens are split in drones, which are sent to the POs for availability check purposes.
The inclusion of gateways (e.g., AND Split, AND Combine, XOR Split according to the BPMN) enables the analyst to represent both successful and failed activities resulting in different end states. An activity’s failure is taken into consideration either stochastically (e.g., exponentially distributed) or by the underlying (physical object) state model. The latter establishes the modelling of interdependencies among activities. Considering a computer network, a PC failure may only affect a limited number of activities whereas the failure of a major component (e.g., a router) may cause the simultaneous unavailability of many BP activities for internet service providers in business-to-business processes (i.e., services). Herewith, the proposed BPM risk analysis approach is a two-stage procedure: First, a conventional BPM (e.g., Boolean state chart, stochastic transitions) is built up; second, physical objects are selected and linked.

### A.1.1 Business process modelling using BPMN

The representation of BP by BPMN is the first step in this IS modelling approach. As this library consists most of the BPMN objects (see Table A.1.1 & Table A.1.1), this is directly done by drag and drop in AnyLogic™. Note, this documentation doesn’t include a detailed description of the BPMN specifications. For a full length documentation of this concept, the authors refer to the BPMN homepage.\(^1\)

Every business process model consists at least of one StartEvent, one Activity and one EndEvent (see Figure A.3). To logically associate the BPMN icons with each other, they have to be connected by so called connectors. The resulting chain of activities then represents the business process. When simulating this process, the AnyLogic engine generates entities (tokens, similar to a Petri-net simulation model) at each StartEvent. This entities then pass trough the process as if they were taws.

---

\(^1\)www.bpmn.org
### Table A.1: Library Objects

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Activity Icon]</td>
<td>Activity</td>
<td>This is the main BPMN object. It represents any activity in a process. Tokens passing through activities are representing the performing of this activity.</td>
</tr>
<tr>
<td>![Activity-Message-In Icon]</td>
<td>Activity-Message-In</td>
<td>An instance of the activity which has the additional capability of receiving messages from other activities</td>
</tr>
<tr>
<td>![Activity-Message-Out Icon]</td>
<td>Activity-Message-Out</td>
<td>An instance of the activity which has the additional capability of sending messages to other activities</td>
</tr>
<tr>
<td>![EndEvent Icon]</td>
<td>EndEvent</td>
<td>This Event represents an end-state of a process. A single process can have multiple EndEvents</td>
</tr>
<tr>
<td>![GatewayAND-Combine Icon]</td>
<td>GatewayAND-Combine</td>
<td>This gateway is able to connect two threats of one task (token). It combines only tokens with the same ID</td>
</tr>
<tr>
<td>![GatewayAND-Split Icon]</td>
<td>GatewayAND-Split</td>
<td>This gateway splits a single task into two threats. Thereby there is generated a copy of the original token with the same ID</td>
</tr>
<tr>
<td>![GatewayXOR-Split Icon]</td>
<td>GatewayXOR-Split</td>
<td>The XOR is the most common gateway. It leaves the token either to the right or to the left side, but never to both</td>
</tr>
<tr>
<td>![PhysicalObject Icon]</td>
<td>PhysicalObject</td>
<td>All physical object are modelled by this generic state-machine. It has a queue and is able to represent a pool of physical objects</td>
</tr>
<tr>
<td>![StartEvent Icon]</td>
<td>StartEvent</td>
<td>The StartEvent is denotes the beginning of a process. It mainly generates tokens in stochastic time-intervals</td>
</tr>
<tr>
<td>![StartEvent-Message Icon]</td>
<td>StartEvent-Message</td>
<td>This start event is triggered by messages. It releases a token for each arriving message</td>
</tr>
<tr>
<td>![VirtualTokenRouter (VTR) Icon]</td>
<td>VirtualTokenRouter (VTR)</td>
<td>The routing of sub-tokens and drones is handled by the virtual token-router. Every model should consist of only one VTR</td>
</tr>
</tbody>
</table>
Table A.2: Special Objects (bound to this library)

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SubActivity</td>
<td>In order to increase the clarity of a business process model, the processes can be structured in sub-processes (SubActivity)</td>
</tr>
<tr>
<td></td>
<td>Swimlane</td>
<td>Every process should be encapsulated in a swimlane. Every swimlane represents an agent regarding the applied agent-based modelling approach. Swimlanes can easily be replicated to represent identical agents.</td>
</tr>
</tbody>
</table>

A.2 com.xj.anylogic

The Java based simulation software AnyLogic comprises of many classes. They are all extensively discussed in the AnyLogic user Manual. Only two of the classes are mentioned here:

A.2.1 Engine

Class Hierarchy:

- java.lang.Object
  - com.xj.anylogic.Engine

Core AnyLogic class that implements thread management. One instance per Java VM is created automatically by the framework. For more Details, please see the AnyLogic user manual.

A.2.2 ActiveObject

Class Hierarchy:

- java.lang.Object
  - com.xj.anylogic.Func
    - com.xj.anylogic.ActiveObject
A.3. The Library Package (bpram)

Besides the "Engine" the ActiveObject is the main and most important class within AnyLogic. It includes numerous methods which all together makes the ActiveObject a perfect default simulation object. For more Details, please see the AnyLogic user manual.

A.3 The Library Package (bpram)

The core objects of the simulation method developed and presented within this thesis are compiled in the package called bpram. It includes Java objects for all major BPMN elements, which can be used by drag and drop within the AnyLogic environment. This section summarizes the most important features of the major bpram classes by declaring the class hierarchy, the field summary and the method summary.

A.3.1 Activity

Class Hierarchy:

```
- ...
- com.xj.anylogic.ActiveObject
  - bpram.basics.ActiveObjectExtended
    - bpram.Activity
```

This is the main BPMN object. It represents any activity in a process. Tokens passing trough activities are representing the performing of this activity.

The activity object is designed generically to represent all kind of different activities. In general, every activity can either succeed or fail, but for design reasons, the activity object can also be used just as a placeholder. In this case (activityMode=Simple Delay) the activity is executed always successful, and it takes a specific time (delayValue) to realise the activity goal. A little bit more sophisticated is to operate the activity in the Availability Check mode. In this mode, the activity is only performed successfully, if all necessary PhysicalObjects are available.
Figure A.2: Class diagramm of the main bpram objects activity and physical object

Table A.3: Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>_activityMode</td>
</tr>
</tbody>
</table>

This parameter is encapsulating the parameter from the properties window because we need them for the inheritance since the params from the properties window are not inherited (as the structure of the active obj too!)
### Table A.3 - continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>_delayValue</td>
</tr>
<tr>
<td></td>
<td>This parameter is capsulating the parameter from the properties window because we need them for the inheritance since the params from the properties window are not inherited (as the structure of the active obj too!)</td>
</tr>
<tr>
<td>int</td>
<td>_irregularity</td>
</tr>
<tr>
<td></td>
<td>This parameter is capsulating the parameter from the properties window because we need them for the inheritance since the params from the properties window are not inherited (as the structure of the active obj too!)</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>_organisationalUnit</td>
</tr>
<tr>
<td></td>
<td>This parameter is capsulating the parameter from the properties window because we need them for the inheritance since the params from the properties window are not inherited (as the structure of the active obj too!)</td>
</tr>
<tr>
<td>int</td>
<td>_resourceMode</td>
</tr>
<tr>
<td></td>
<td>This parameter is capsulating the parameter from the properties window because we need them for the inheritance since the params from the properties window are not inherited (as the structure of the active obj too!)</td>
</tr>
<tr>
<td>int</td>
<td>actID</td>
</tr>
<tr>
<td></td>
<td>ID (identity number) of the activity</td>
</tr>
<tr>
<td>static int</td>
<td>actIDPool</td>
</tr>
<tr>
<td></td>
<td>This is a pool of unique identity numbers.</td>
</tr>
<tr>
<td>int</td>
<td>activityMode</td>
</tr>
<tr>
<td></td>
<td>(int), Default value is &quot;Simple Delay&quot;.</td>
</tr>
<tr>
<td>static int</td>
<td>AVAILABILITY_CHECK_MODE</td>
</tr>
<tr>
<td></td>
<td>Used in this and the derived classes Activity_Message_Out and Activity_Message_In in the parameter activityMode or _activityMode respectively</td>
</tr>
</tbody>
</table>
### Table A.3 - continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static int</strong></td>
<td>BROADCAST</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>When the subToken returns from the availability check, it passes the collected information to its token, then it is destroyed.</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>Delay</td>
</tr>
<tr>
<td><strong>double</strong></td>
<td>delayValue (double), Default value is &quot;10&quot;.</td>
</tr>
<tr>
<td><strong>protected java.util.Vector</strong></td>
<td>destinationResources</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>enterSubToken</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>exitSubToken</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. Port</strong></td>
<td>inPort</td>
</tr>
<tr>
<td><strong>int</strong></td>
<td>irregularity (int), Default value is &quot;1&quot;.</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>matchQ</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>BROADCAST</strong></td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>Used in derived classes Activity_Message_Out in parameter transmitMessage and called with the sendMessage() method.</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterprise_library.</strong></td>
<td>combine</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. enterSubToken</strong></td>
<td>This object is a &quot;token teleporter element&quot;.</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. exitSubToken</strong></td>
<td>This object is a &quot;token teleporter element&quot;.</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. inPort</strong></td>
<td>This is the default inPort of AnyLogic</td>
</tr>
<tr>
<td><strong>int</strong></td>
<td>This is the default inPort of AnyLogic</td>
</tr>
<tr>
<td><strong>com.xj.anylogic. matchQ</strong></td>
<td>After the token has passed the subtokenCreator, it is placed in the matchQ.</td>
</tr>
</tbody>
</table>
### Table A.3 - continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| java.lang.String | organisationalUnit  
(string), IMPORTANT: The inserted string has to be written in between two double quotes ("O"). |
| com.xj.anylogic. Port | outPort  
This is the default outPort of AnyLogic |
| com.xj.anylogic. lib.enterprise_library. Queue | queue  
This queue prevents a proper handling of concurrent tokens and has the role of a buffer element. |
| com.xj.anylogic. lib.enterprise_library. Queue | queueDelay  
This queue prevents, that concurrent tokens are processed one by one with a FIFO algorithm. |
| int | resourceMode  
(int), Default value is "Synchronous Mode". |
| com.xj.anylogic. lib.enterprise_library. SelectOutput | selectOutput  
This object checks the activityMode of the Function. |
| static int | SIMPLE_DELAY_MODE  
Used in this and the derived classes Activity_Message_Out and Activity_Message_In in the parameter activityMode or _activityMode respectively |
| static int | SINGLECAST  
Used in derived classes Activity_Message_Out in parameter transmitMessage and called with the sendMessage() method. |
| com.xj.anylogic. lib.enterprise_library. Split | subtokenCreator  
If the activityMode is set to Availability Check, the status of the physicalObject, which are needed to perform this activity, have to be inspected. |

**End of Table A.3**

### Table A.4: Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>protected</td>
<td><code>_ActivityExit(Token token)</code></td>
</tr>
<tr>
<td><em>void</em></td>
<td>This method just provokes an entry into the logfile.</td>
</tr>
<tr>
<td><em>void</em></td>
<td><code>ActivityEnteredQueue()</code></td>
</tr>
<tr>
<td><em>void</em></td>
<td>This method is called when a token enters the inner queue of the activity.</td>
</tr>
<tr>
<td><em>void</em></td>
<td><code>ActivityExitCombine()</code></td>
</tr>
<tr>
<td><em>void</em></td>
<td>This method is called when a Token passes the exit port of the inner delay object at the moment, it triggers only the <code>_ActivityExit(bpram.Token)</code> method which provokes an entry into the logfile.</td>
</tr>
<tr>
<td><em>void</em></td>
<td><code>ActivityExitDelay()</code></td>
</tr>
<tr>
<td><em>void</em></td>
<td>This method is called when a Token passes the exit port of the inner delay object at the moment, it triggers only the <code>_ActivityExit(bpram.Token)</code> method which provokes an entry into the logfile.</td>
</tr>
<tr>
<td><em>void</em></td>
<td><code>activityOnExitCopy()</code></td>
</tr>
<tr>
<td><em>void</em></td>
<td>This method is called before the SubToken leaves the inner object &quot;subtokenCreator&quot;.</td>
</tr>
</tbody>
</table>

1. Triggering the SubToken. `setSourceActivity` (com.xj.anylogic. ActiveObject) method, it provides that the subToken "knows" to which activity it has to return after the availability check.

2. Triggering the `bpram.SubToken setDestination-Resources` method, it passes the destinationResources vector of the activity to the SubToken.

3. Triggering the SubToken `setMode(int)` method, it passes the `_resourceMode` of the activity to the SubToken.
### Table A.4 - continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>create()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>com.xj.anylogic.animation.Group</td>
<td>createIcon()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>com.xj.anylogic.animation.Group</td>
<td>createStructure()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>void</td>
<td>destroy()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>int</td>
<td>getActID()</td>
</tr>
<tr>
<td></td>
<td>Returns the (ID) identity number of the activity</td>
</tr>
<tr>
<td>java.util.Vector</td>
<td>getDestinationResources()</td>
</tr>
<tr>
<td></td>
<td>This method returns the destinationResources of the activity.</td>
</tr>
<tr>
<td>protected boolean</td>
<td>getGlobalAvail(Token token)</td>
</tr>
<tr>
<td></td>
<td>This boolean method returns &quot;true&quot;, if all PhysicalObjects in the destinationResources are available.</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getType()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>void</td>
<td>initActivity()</td>
</tr>
<tr>
<td></td>
<td>This method is called to initialise a new activity object</td>
</tr>
<tr>
<td>protected void</td>
<td>initDPool()</td>
</tr>
<tr>
<td></td>
<td>This method is called during the initializing of the activity object.</td>
</tr>
<tr>
<td>protected void</td>
<td>initParams()</td>
</tr>
<tr>
<td></td>
<td>This method merges the internal and the external parameters.</td>
</tr>
<tr>
<td>static void</td>
<td>main(java.lang.String[] args)</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>void</td>
<td>onEngineCreate()</td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
<tr>
<td>void</td>
<td>setDestinationResources(java.util.Vector v)</td>
</tr>
<tr>
<td></td>
<td>This method is called by the Wizard with the savePhysicalsForActivity method in order to initialise the destinationResources.</td>
</tr>
</tbody>
</table>
### Table A.4 - continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void</code></td>
<td><code>setParams(int activityMode, int resourceMode, double delayValue, java.lang.String organisationalUnit, int irregularity)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the parameters</td>
</tr>
<tr>
<td><code>protected void</code></td>
<td><code>startup()</code></td>
</tr>
<tr>
<td></td>
<td>AnyLogic method</td>
</tr>
</tbody>
</table>

End of Table A.4

#### A.3.1.1 Activity_Message_In

**Class Hierarchy:**

```plaintext
¬ ...  
  ¬ bpram.basics.ActiveObjectExtended  
  ¬ bpram.Activity  
  ¬ bpram.Activity_Message_In
```

This object is derived from the Activity class, and has basically the same functionality. In contrast to the standard Activity, it has an additional messagePort. Through this port, the incoming messages from other activities (Activity_Message_Out) are handled.

In fact, the Activity_Message_In is always triggered by another activity usually from another swimlane. This triggering is realised by sending messages. In contrast to the Activity_Message_Out object, the Activity_Message_In can only receive messages, but not send them. The arriving messages, provoke the execution of the Activity.

#### A.3.1.2 Activity_Message_Out

**Class Hierarchy:**

```plaintext
¬ ...  
  ¬ bpram.basics.ActiveObjectExtended  
  ¬ bpram.Activity
```
A.3. The Library Package (bpram)

\[ \texttt{bpram}\textunderscore\texttt{Activity\_Message\_Out} \]

This object is derived from the activity class, and has basically the same functionality. In contrast to the standard activity, it has an additional message port. Through this port, the outgoing messages to other activities (Activity\_Message\_In) or StartEvent\_Message are handled.

In fact, the Activity\_Message\_Out is used to trigger other activities usually in other swimlanes. This triggering is realised by sending messages. In contrast to the Activity\_Message\_In object, the Activity\_Message\_Out can only send messages, but not receive them. The sent messages provoke the execution of the triggered activities.

A.3.2 PhysicalObject

Class Hierarchy:

\[ \ldots \]
\[ \texttt{com.xj.anylogic}\textunderscore\texttt{ActiveObject} \]
\[ \texttt{bpram}\textunderscore\texttt{basics}\textunderscore\texttt{ActiveObjectExtended} \]
\[ \texttt{bpram}\textunderscore\texttt{PhysicalObject} \]

In order to model the state of physical objects, a generic physical object (PO) class was designed (see section 3.8). The main concept of this simulation approach lies in the interplay between the business processes and these PO. If a PO is used to perform a specific activity within a process, it has to be "occupied" by this activity. This occupation is realised by sending drones to the PO, which at the same time check the availability of the PO. If a PO in principle is available, but occupied, the drone has to wait in a FIFO queue until it's it can be processed. But if the PO is not available, which is simulated by a

![Diagram of a very simple business process](image)

**Figure A.3:** *Illustration of a very simple business process*
stochastic expiration of the mean-time to failure (MTTF), the drone returns immediately to its source activity with the information of the "not available" state of the PO.

A.3.3 Events

![Class diagramm of the implemented bpram events](image-url)
A.3. The Library Package (bpram)

A.3.3.1 StartEvent

Class Hierarchy:

```
  ...
  ∨ com.xj.anylogic.ActiveObject
  ∨ bpram.basics.ActiveObjectExtended
  ∨ **bpram.StartEvent**
```

The start event produces new tokens and outputs them through the output port. This class is the base class for StartEvent_Message and StartEvent Timer (not yet implemented). The start event message acts the same way as the start event itself but it is triggered by an incoming message. See class description of StartEvent_Message for more details.

A.3.3.2 StartEvent_Message

Class Hierarchy:

```
  ...
  ∨ com.xj.anylogic.ActiveObject
  ∨ bpram.basics.ActiveObjectExtended
  ∨ **bpram.StartEvent_Message**
```

A.3.3.3 EndEvent

Class Hierarchy:

```
  ...
  ∨ com.xj.anylogic.ActiveObject
  ∨ bpram.basics.ActiveObjectExtended
  ∨ **bpram.EndEvent**
```

This object disposes incoming tokens. It is used as an end point of the entity flow in a business process. At this point, the contents of the tokens (data) is posted into a logfile. The EndEvent represents either a successful or a failure end of a process.
### A.3.3.4 IntermediateEvent

**Class Hierarchy:**

- ...  
- bpram.basics.ActiveObjectExtended  
- bpram.Activity  
- bpram.IntermediateEvent

This object is derived from the Activity class, and has basically the same functionality. In contrast to the standard Activity, it has an additional messagePort. Through this port, the incoming messages from other activities (Activity_Message_Out) are handled.

In fact, the Activity_Message_In is always triggered by another activity usually from another swimlane. This triggering is realised by sending messages. In contrast to the Activity_Message_Out object, the Activity_Message_In can only receive messages, but not send them. The arriving messages, provoke the execution of the Activity

### A.3.4 Gateways

![Class diagramm of the implemented bpram gateways](image)

**Figure A.5:** *Class diagramm of the implemented bpram gateways*
A.3.4.1 GatewayAND_Combine

**Class Hierarchy:**

```
  \node{...};
  \node{com.xj.anilogic.ActiveObject};
  \node{bpram.basics.ActiveObjectExtended};
  \node{bpram.GatewayAND_Combine};
```

Gateways are logical operators for the token-flow control. The AND combine gateway assembles tokens which represent different "threats" from the same "task". Only tokens with the same tokenID can be combined which implies that there have to precede an AND split gateway previously in the process.

A.3.4.2 GatewayAND_Split

**Class Hierarchy:**

```
  \node{...};
  \node{com.xj.anilogic.ActiveObject};
  \node{bpram.basics.ActiveObjectExtended};
  \node{bpram.GatewayAND_Split};
```

The AND split gateway separates tokens which represent different "threats" from the same "task". The split tokens get both the same tokenID which implies that the two threats belong to the same task.

A.3.4.3 GatewayXOR_Split

**Class Hierarchy:**

```
  \node{...};
  \node{com.xj.anilogic.ActiveObject};
  \node{bpram.basics.ActiveObjectExtended};
```
The XOR Split guides the token either to one or to the other port, depending on the mode of the gateway (GLOBAL_AVAILABILITY, maxLoops or probability).

A.3.5 Entities

As described above, the simulation model is based on an entity flow model. The entity class is a fundamental class of the AnyLogic\textsuperscript{TM} Enterprise Library for all message objects that travel between the activities and physical objects of this library. An entity may represent an entity in its conventional meaning (product, order, customer, data packet) or a resource unit (operator, machine, critical section). Entities as tokens are generated at StartEvent objects, and then flow through the business process being modeled, get processed by the activities, and then exit the system through the EndEvent object. Entities are also used within the physical objects as a resource unit. There, they are generated at resource objects inside the PO, are seized by drones, released and returned to the resource object (see Figure ??). An entity may contain other entities and so on to any desired depth. The contained entities are stored in the field contents of type Vector. The resource units seized by the entity are stored in the Vector resources. Within this library, several subclasses of entities are defined such as the tokens, subTokens, drones and different message entities (see below)

A.3.5.1 Token

The token is the main entity used for the business process simulation. It is released by the StartEvent, flows through the connected activity objects collecting all sort of performance data, and then exits the system through the an EndEvent.

Class Hierarchy:

```java
- java.lang.Object
  - com.xj.anylogic.lib.enterprise_library.Entity
    - bpram.Token
```
### A.3. The Library Package (bpram)

#### A.3.5.2 SubToken

**Class Hierarchy:**

- java.lang.Object
  - com.xj.anylogic.lib.enterprise_library.Entity
    - bpram.SubToken

#### A.3.5.3 Drone

**Class Hierarchy:**

- java.lang.Object
  - com.xj.anylogic.lib.enterprise_library.Entity
    - bpram.Drone
A.4 bpram.basics

Figure A.7: Class diagramm of the implemented basic bpram objects

A.4.1 ActiveObjectExtended

Class Hierarchy:

- ... com.xj.anylogic.Func
- ... com.xj.anylogic.ActiveObject
- ... bpram.basics.ActiveObjectExtendedt
All Implemented Interfaces: java.awt.datatransfer.Transferable, java.lang.Runnable


A.4.2 CombineExtended

Class Hierarchy:

- ...
- com.xj.anynlogc.ActiveObject
  - bpram.basics.ActiveObjectExtended
    - bpram.basics.CombineExtended

All Implemented Interfaces: java.awt.datatransfer.Transferable, java.lang.Runnable

A.4.3 ConcurrSubTokenHandler

Class Hierarchy:

- ...
- com.xj.anynlogc.ActiveObject
  - bpram.basics.ActiveObjectExtended
    - bpram.basics.ConcurrSubTokenHandler

All Implemented Interfaces: java.awt.datatransfer.Transferable, java.lang.Runnable

A.4.4 MatchQExtended

Class Hierarchy:
All Implemented Interfaces: java.awt.datatransfer.Transferable, java.langRunnable

A.4.5 StatsAnalyzer

Class Hierarchy:

All Implemented Interfaces: java.awt.datatransfer.Transferable, java.langRunnable

A.4.6 SubActivity

Class Hierarchy:

All Implemented Interfaces: java.awt.datatransfer.Transferable, java.langRunnable

...
A.4.7  SyncSubTokenHandler

Class Hierarchy:

- ...
  - com.xj.anylogic.ActiveObject
    - bpram.basics.ActiveObjectExtended
      - bpram.basics.SyncSubTokenHandler

All Implemented Interfaces:  java.awt.datatransfer.Transferable, java.langRunnable

A.4.8  TokenDuplicator

Class Hierarchy:

- ...
  - com.xj.anylogic.ActiveObject
    - bpram.basics.ActiveObjectExtended
      - bpram.basics.TokenDuplicator

All Implemented Interfaces:  java.awt.datatransfer.Transferable, java.langRunnable

A.4.9  VirtualTokenRouter

Class Hierarchy:

- ...
  - com.xj.anylogic.ActiveObject
    - bpram.basics.ActiveObjectExtended
      - bpram.basics.VirtualTokenRouter
The VirtualTokenRouter (VRT) is responsible for the correct routing between the business process and the physical objects. Subtokens enter the VRT from different activities of the business process. The VRT creates the appropriate numbers of Drones according to the parameters of the Subtoken. The Drones are then send to the physical objects. In the end the VRT collects all the Drones and adds them to the Subtoken which is send back to the correct activity.

**All Implemented Interfaces:** java.awt.datatransfer.Transferable, java.langRunnable
Appendix B

Terminology

In common speech, the terms risk, reliability, failure etc. are used inconsistent and nonuniform. As these terms have strictly technical meaning within this thesis, it is appropriate to give an overview about the terminology of the most important words and phrases.

Risk Analysis: A process of systematically collecting information about an item in order to identify and estimate risk [11].

Risk Evaluation: A process of comparing the estimated risks [11]. This may include the ranking of risks which helps decision makers to allocate resources for possible countermeasures (risk treatment).

Risk Assessment: The "overall process of risk analysis and risk evaluation" [11]. It may include quantitative (probabilistic) or qualitative risk estimation.

Probabilistic Risk Assessment (PRA): Up to now, PRA has primarily been a process to assessing and quantify risk through the reliability of an item (e.g. a technical system), running either in series or parallel or both, where in general, risk assessment also includes quantitative risk estimation techniques.

Risk Management: "A coordinated activity to direct and control an organization with regard to risk. It generally includes risk assessment, risk treatment, risk acceptance and risk communication" [11].
Reliability \((R, R(t))\) "Probability that the required function of an item will be provided under given conditions for a given time interval" [139]. Reliability is a characteristic of an item, generally designated by \(R\) for the case of a fixed mission and \(R(t)\) for a mission with \(t\) as a Parameter. A qualitative definition, focused on ability, is also possible. Reliability gives the probability that no operational interruption will occur during a stated mission of an item, say of duration \(T\). This does not mean that redundant parts may not fail, such parts can fail and be repaired. Thus, the concept of reliability applies for nonrepairable as well as for repairable items. Should \(T\) be considered as a variable \(t\), the reliability function is given by \(R(t)\). If \(\tau\) is the failure-free time, distributed according to \(F(t)\), with \(F(0) = 0\), then \(R(t) = Pr(\tau > t) = 1 - F(t)\), where \(F(t)\) is the distribution and \(f(t) = \frac{dF(t)}{dt}\) the density function of the random variable \(\tau\).

Availability: is the "probability that the item is in a state to perform the required function at a given instant of time" [139]. The term item stands for a structural unit of arbitrary complexity. Assuming renewal for the whole item, the asymptotic & steady-state value of the point availability (\(PA\)) can be expressed by \(PA = \frac{MTTF}{MTTF + MTTR}\). \(PA\) is also the asymptotic & steady-state value of the average availability \(AA\) (often given as availability \(A\)).

Failure: "Termination of the ability to perform the required function" [139]. Failures should be considered (classified) with respect to the mode, cause, effect, and mechanism. The cause of a failure can be intrinsic (early failure, failure with constant failure rate, wearout) or extrinsic (systematic failures, i.e. failures resulting from errors or mistakes in design, production, or operation which are deterministic and has to be considered as defects). The effect (consequence) of a failure is often different if considered on the directly affected item or on a higher level. A failure is an event appearing in time (randomly distributed), in contrast to a fault which is a state.

Failure Rate \((\lambda(t))\) Limit, if it exists, of the conditional probability that the failure occurs within time interval \((t, t + \delta t)\), when \(\delta t \to 0\), given that the item was new at \(t = 0\) and did not fail in the interval \((0, t)\). At system level, \(\lambda_S(t)\) is used. The failure rate applies in particular for nonrepairable items. In this case, if \(\tau\) is the item failure-free time, with distribution function \(F(t) = Pr(\tau < t)\), with \(F(0) = 0\) and density \(f(t)\), the failure rate \(\lambda(t)\) follows with \(R(t) = 1 - F(t)\) as follows:
\[ \lambda(t) = \lim_{\delta t \to 0} \frac{1}{\delta t} \Pr\{t < \tau \leq t + \delta t | \tau > 0\} = \frac{f(t)}{1 - F(t)} = -\frac{dR(t)/dt}{R(t)} \] (B.1)

Considering \( R(0) = 1 \), Equation B.1 yields \( R(t) = e^{\int_0^t \lambda(x)dx} \) and thus \( R(t) = e^{-\lambda t} \). This important result characterizes the memoryless property of the exponential distribution \( F(t) = 1 - e^{-\lambda t} \), for \( \lambda(t) = \lambda \). Only for \( \lambda(t) = \lambda \) one can estimate the failure rate \( \lambda \) by \( \lambda = \frac{k}{T} \) where \( T \) is the given (fixed) cumulative operating time and \( k > 0 \) the total number of failures during \( T \). However, considering Equation B.1, the failure rate can be defined also for repairable items which are as-good-as-new after repair (restoration), taking instead of \( t \) the variable \( x \) starting by \( x = 0 \) at each repair (as for interarrival times). This is important when investigating repairable Systems, e.g. with constant failure & repair rates. If a repairable system cannot be restored to be as-good-as-new after repair (with respect to the state considered), i.e. if at least one element with time dependent failure rate has not been renewed at every repair, failure intensiy \( z(t) \) has to be used. It is thus important to distinguish between failure rate \( \lambda(t) \) and failure intensity \( z(t) \) or intensity \( h(t) \) or \( m(t) \) for a renewal or Poisson process). \( z(t), h(t), m(t) \) are unconditional densities and differ basically from \( \lambda(t) \) which is a conditional density. This distinction is important also for the case of a homogeneous Poisson process, for which \( z(t) = h(t) = m(t) = \lambda \) holds for the intensity and \( \lambda(x) = \lambda \) holds for the interarrival times (\( x \) starting by \( 0 \) at each interarrival time). To reduce ambiguities, force of mortality has been suggested for \( \lambda(t) \).

**Fault:** "State characterized by an inability to perform the required function due to an internal reason" [139]. A fault is a state and can be a defect or a failure, having thus as possible cause an error (for defects or systematic failures) or a failure mechanism (for failures).

**Maintenance:** "defines the set of activities performed on an item to retain it in or to restore it to a specified state. Maintenance is thus subdivided into preventive maintenance, carried out at predetermined intervals to reduce wearout failures, and corrective maintenance, carried out after failure recognition and intended to put the item into a state in which it can again perform the required function. Aim of a preventive maintenance is also to detect and repair hidden failures, i.e. failures in redundant elements not identified at their occurrence. Corrective maintenance is also known as repair, and can include any or all of the following steps: recognition, isolation (localization & diagnosis), elimination (disassembly, replace, reassembly), checkout. Repair is used hereafter as
a synonym for restoration. To simplify calculations, it is generally assumed that the element in the reliability block diagram for which a maintenance action has been performed is as-good-as-new after maintenance. This assumption is valid for the whole equipment or system in the case of constant failure rate for all elements which have not been repaired or replaced.

Maintainability is a characteristic of an item, expressed by the probability that a preventive maintenance or a repair of the item will be performed within a stated time interval for given procedures and resources (skill level of personnel, spare Parts, test facilities, etc.). From a qualitative point of view, maintainability can be defined as the ability of an item to be retained in or restored to a specified state. The expected value (mean) of the repair time is denoted by $MTTR$ (mean time to repair), that of a preventive maintenance by $MTTPM$. Often used for unscheduled removals is also $MTBUR$. Maintainability has to be built into complex equipment or Systems during design and development by realizing a maintenance concept. Due to the increasing maintenance cost, maintainability aspects have grown in importance. However, maintainability achieved in the field largely depends on the resources available for maintenance (human and material), as well as on the correct installation of the equipment or system, i.e. on the logistic support and accessibility" [139].

**Safety**  Ability of the item to cause neither injury to persons, nor significant material damage or other unacceptable consequences. Safety expresses freedom from unacceptable risk of harm. In practical applications, it is useful to subdivide safety into accident prevention (the item is safe working while it is operating correctly) and technical safety (the item has to remain safe even if a failure occurs). Technical safety can be defined as the probability that the item will not cause injury to persons, significant material damage or other unacceptable consequences above a given (fixed) level for a stated time interval, when operating under given conditions. Methods and procedures used to investigate technical safety are similar to those used for reliability analysis, however with emphasis on fault/failure effects.

**Security**  "is the condition of being protected against danger or loss. In the general sense, security is a concept similar to safety. The nuance between the two is an added emphasis on being protected from dangers that originate from outside. Individuals or actions that encroach upon the condition of protection are responsible for the breach of security."¹