Master thesis
Information Systems

Business process modeling
with a focus on the aspects of structured design
and Theory of Gestalt

June 10, 2008
Karin Margarete Grünauer
karin.grunauer@gmx.at

Supervisor: Alfred Holl
Examinator: Sara Eriksén

Växjö Universität
Matematiska och Systemtekniska Institutionen (MSI)
This master thesis in information systems has been written at the School of Mathematics and System Engineering (MSI) at Växjö university.

I would like to thank Alfred Holl and Sara Eriksén for their support, feedback and great ideas during the creation of this thesis.

I also want to thank my American friends Lisa Davis and Renee Gang who helped me proof reading my master thesis.

Finally I want to thank my parents, my grandma, and my brother who always trusted in my work and supported my plans, not only financially.

Växjö, June 2008

Karin Margarete Grünauer
Abstract/Motivation

Business processes represent a transformation: input is transformed and output with a higher value, higher usefulness and effectiveness for the recipient is created. Depending on the business strategy and the process goals, a business process creates profit for the company. This is the reason why business process modeling (BPM) is so popular nowadays. Every company wants to optimize their processes for creating higher effectiveness and of course larger profits. Furthermore, business processes are normally modeled with the aim that parts of their functionality are supported or implemented by a workflow management system.

As it seems is there a close relation between Computer science (CS) and Information systems (IS). But this is not the fact. For modeling business processes, a model type which was originally developed for software engineering is nowadays used for business process modeling (BPM). This development led to a big jumble of graphical BPM notations: Sometimes graphical notations are used which are not adequate for BPM.

By a genealogical comparison and a comparison using a standardized example the different graphical notations will be evaluated with a strengths and weaknesses profile. The found analogies of those notations and control flow modeling are presented in Table 19 in the end of Section 3.

A high-quality business process model is a good basis for workflow models which are derived from the refinement of business process models.

Therefore, is it very distressing to see that now, nearly 40 years after the introduction of structured software design, structured design for business process modeling is still being neglected. Figure 1 shows a business process model of a german organization, published by Sueddeutsche Zeitung in April 2008.

Figure 1: Example of the current modeling style (Sueddeutsche Zeitung, April 14, 2008 [online])
It is interesting that none of the current notations for BPM automatically leads to structured models. Even UML allows the freedom of unstructured design. BPM has to adopt the features of structured programming to make business process models more transparent, which simplifies their graphical and verbal documentation as well as their modification and adaptation. Structured business process models would be easier to optimize and the mapping of workflow management systems would become more effective. Any data processing organization should use structured techniques today - why however don’t process-oriented organizations do this?

The current BPM style has to be analyzed and discussed, as well as the approach of structured BPM (SBPM) including its advantages. With the help of a practical example the necessity of the structured approach can be proved.

In order to optimize business processes models they have to be split into partial models to minimize their complexity. Business process decomposition on several levels of abstraction is an essential part in making existent business process models more transparent, evaluable and discussable.

The aspects of Theory of Gestalt are the fundamentals for an approach for a motivated decomposition. Decomposition is an unconscious, creative act of every single model designer and is done with the help of features. A notation which includes the feature aspect would be desirable. For that reason the feature oriented event driven process chain will be presented in the end of Section 4 as an example for an approach which enables the motivated decomposition of business process models on several levels of abstraction.

Key words

Process model, behavioral model, business process model, control flow models, data flow models, object orientation, object oriented models, structured modeling, structured design, event driven process chain, extended event driven process chain, object oriented event driven process chain, UML activity diagram, flowcharts, Nassi-Shneiderman, Theory of Gestalt, business process, business process modeling (BPM), modeling languages, data flow diagrams (SA, SSA), flow diagrams (SADT), IDEF0, IDEF3, Petri net, swim lane, PROMET, BP diagram, block diagram, control flow chart, UML, decomposition;
Outline of the thesis

In Section 1 of my thesis I will explain and define the quintessential terms used in business process modeling (BPM) and present the methodology, including problem definition, goals, methods and results of this master thesis.

After this overview, I will describe the different approaches and styles in BPM. The main focus of this section explores the current style of BPM and the demand for structured design. An example will illustrate the defects caused by the current unstructured modeling style and furthermore show how an unstructured diagram can be transformed into a structured business process model.

Section 3 is dedicated to the common graphical notations in BPM. I created a genealogical tree with the most common business process notations, each sorted by release date, in which one can see the influence of the different relations. A standardized order example will be illustrated with every described graphical notation for comparison and evaluation reasons. In doing so, I found ideas of improvements for some notations which are described in the end of the description of every graphical notation. Furthermore will every graphical notation be evaluated with the help of a strengths and weaknesses profile.

Next to the question of whether object orientation and UML are applicable to business process modeling, I will present a table with the analogies between the different graphical notations in the end of this section.

In Section 4, I will discuss the decomposition of business processes using Theory of Gestalt specifically focusing on how Theory of Gestalt can positively influence business process models by making them more transparent and discussable, particularly in reference to decomposition on several levels of abstraction.

The conclusion of my work can be found in Section 5 and the bibliography follows in Section 6.
## Table of contents

Abstract/Motivation ................................................................................................................. 3
Key words ................................................................................................................................. 4
Outline of the thesis ..................................................................................................................... 5
Table of contents .......................................................................................................................... 6

1. Introduction ............................................................................................................................... 9
   1.1. Methodology ....................................................................................................................... 10
   1.1.1. Problem definitions ......................................................................................................... 11
   1.1.2. Goals .............................................................................................................................. 12
   1.1.3. Methods ......................................................................................................................... 13
   1.1.4. Results ........................................................................................................................... 14

1.2. Definitions/Terminology ........................................................................................................ 15
   1.2.1. Business process ........................................................................................................... 15
   1.2.2. Workflow ....................................................................................................................... 16
   1.2.3. Model and reality ......................................................................................................... 16
   1.2.4. Modeling notations ....................................................................................................... 18
   1.2.5. Process .......................................................................................................................... 18
   1.2.6. Control flow .................................................................................................................. 18
   1.2.7. Data flow ...................................................................................................................... 19

2. Structured Business Process Modeling (SBPM) ......................................................................... 20
   2.1. Current situation/The unstructured modeling style in BPM ............................................. 21
   2.2. Demonstration of the defects caused by unstructured BPM style ..................................... 22
   2.3. Requirements to structured BPM ..................................................................................... 28

3. Graphical notations of business process models ......................................................................... 29
   3.1. Genealogical tree of graphical notations .......................................................................... 30
   3.2. Distinction of the different graphical notations ............................................................... 32
   3.2.1. Object orientation in BPM .......................................................................................... 33
   3.2.2. UML and business process modeling .......................................................................... 33
   3.2.3. Standardized Example ............................................................................................... 34
   3.2.4. Evaluation of the graphical notations .......................................................................... 35
   3.3. Control flow chart .............................................................................................................. 37
   3.3.1. Notation of control flow charts .................................................................................... 38
   3.3.2. Demonstration of a flow chart diagram ....................................................................... 39
   3.3.3. Evaluation of the control flow chart notation .............................................................. 40
   3.3.4. Idea for improvement of control flow diagrams ............................................................ 40
   3.4. Network modeling technique (NMT) .............................................................................. 42
   3.4.1. Event on node network ................................................................................................ 42
   3.4.2. Activity on node network ............................................................................................ 42
   3.4.3. Activity on arc network ................................................................................................ 43
   3.4.4. Activity Table ............................................................................................................... 43
   3.4.5. Time analysis ............................................................................................................... 44
   3.4.6. Demonstration of an activity on arc network .............................................................. 45
   3.4.7. Evaluation of the NMT activity on arc networks notation ........................................... 45
   3.4.8. Important difference to other graphical notations ......................................................... 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Petri net</td>
<td>47</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Notation of Place/Transition-Nets</td>
<td>49</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Demonstration of a Petri net</td>
<td>50</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Evaluation of the P/T Petri nets without marks notation</td>
<td>51</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Idea for improvement of Petri Nets</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>Structure diagram (Nassi Shneidermann)</td>
<td>53</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Notation of Nassi Shneiderman structure diagrams</td>
<td>53</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Demonstration of a Structure Diagram</td>
<td>55</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Evaluation of the Nassi Shneiderman notation</td>
<td>55</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Idea for improvement of Nassi Shneiderman diagrams</td>
<td>56</td>
</tr>
<tr>
<td>3.7</td>
<td>SADT/IDEFO/IDEF3</td>
<td>57</td>
</tr>
<tr>
<td>3.7.1</td>
<td>SADT and IDEFO</td>
<td>57</td>
</tr>
<tr>
<td>3.7.2</td>
<td>SADT/IDEFO notation</td>
<td>57</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Demonstration of an IDEFO-Diagram</td>
<td>59</td>
</tr>
<tr>
<td>3.7.4</td>
<td>IDEF3</td>
<td>59</td>
</tr>
<tr>
<td>3.7.5</td>
<td>Evaluation of the IDEFO/SADT notation</td>
<td>61</td>
</tr>
<tr>
<td>3.8</td>
<td>Data flow diagrams (SA, SSA)</td>
<td>61</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Notation of data flow diagrams</td>
<td>62</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Demonstration of a data flow diagram (DFD)</td>
<td>64</td>
</tr>
<tr>
<td>3.8.3</td>
<td>Evaluation of the data flow diagram notation</td>
<td>65</td>
</tr>
<tr>
<td>3.8.4</td>
<td>Idea for improvement of data flow diagrams</td>
<td>65</td>
</tr>
<tr>
<td>3.9</td>
<td>Swim lane diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.9.1</td>
<td>Notation of Swim lane diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.9.2</td>
<td>Demonstration of a swim lane diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.9.3</td>
<td>Evaluation of the swim lane diagram notation</td>
<td>66</td>
</tr>
<tr>
<td>3.9.4</td>
<td>Idea for improvement of swim lane diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.10</td>
<td>Block diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Notation of block diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Demonstration of a block diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.10.3</td>
<td>Evaluation of the block diagram notation</td>
<td>66</td>
</tr>
<tr>
<td>3.10.4</td>
<td>Idea for improvement of block diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.11</td>
<td>Event-driven process chain</td>
<td>66</td>
</tr>
<tr>
<td>3.11.1</td>
<td>Notation of EPCs</td>
<td>66</td>
</tr>
<tr>
<td>3.11.2</td>
<td>Demonstration of an event driven process chain</td>
<td>66</td>
</tr>
<tr>
<td>3.11.3</td>
<td>Evaluation of the EPC notation</td>
<td>66</td>
</tr>
<tr>
<td>3.11.4</td>
<td>Idea for improvement of EPC diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.12</td>
<td>PROMET diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.12.2</td>
<td>Demonstration of a PROMET diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.12.3</td>
<td>Evaluation of the PROMET notation</td>
<td>66</td>
</tr>
<tr>
<td>3.12.4</td>
<td>Idea for improvement of PROMET diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.13</td>
<td>UML activity diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.13.1</td>
<td>Notation of activity diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.13.2</td>
<td>Demonstration of an activity diagram</td>
<td>66</td>
</tr>
<tr>
<td>3.13.3</td>
<td>Evaluation of the UML activity diagram notation</td>
<td>66</td>
</tr>
<tr>
<td>3.13.4</td>
<td>Idea for improvement of UML activity diagrams</td>
<td>66</td>
</tr>
<tr>
<td>3.14</td>
<td>Extended event driven process chains (eEPC)</td>
<td>66</td>
</tr>
<tr>
<td>3.14.1</td>
<td>Notation of eEPC diagrams</td>
<td>66</td>
</tr>
</tbody>
</table>
3.14.2. Demonstration of an extended EPC ................................................................. 66
3.14.3. Evaluation of the eEPC notation .................................................................. 66
3.14.4. Idea for improvement of eEPC diagrams .................................................... 66

3.15. Business process diagram (BPD) .................................................................. 66
3.15.1. Notation of BP diagrams ........................................................................... 66
3.15.2. Demonstration of a BP diagram ................................................................ 66
3.15.3. Evaluation of the BPD notation ................................................................. 66
3.15.4. Idea for improvement of BP diagrams ....................................................... 66

3.16. Object oriented event driven process chain (oEPC) ....................................... 66
3.16.1. Notation of oEPCs .................................................................................... 66
3.16.2. Demonstration of an oEPC ...................................................................... 66
3.16.3. Evaluation of the oEPC notation .............................................................. 66
3.16.4. Idea for improvement of oEPC diagrams .................................................. 66

3.17. Analogy of the elementary components of control flow models and business process models ........................................................................................................ 66

4. Business process decomposition under the aspects of Theory of Gestalt .......... 66
   4.1. Business process decomposition .................................................................... 66
   4.2. Theory of Gestalt .......................................................................................... 66
   4.3. Theory of Gestalt, processes, and their decomposition .................................. 66
   4.4. Business process modeling and decomposition ............................................. 66
   4.4.1. Comparison with normalized data modeling .............................................. 66
   4.5. Target state of business process modeling ................................................... 66
   4.5.1. Features .................................................................................................... 66
   4.5.2. Relations between features and events ....................................................... 66
   4.6. Business process representation ................................................................... 66
   4.6.1. Evolution of feature ................................................................................ 66

5. Conclusion .......................................................................................................... 66

6. Bibliography & Indexes ....................................................................................... 66
   6.1. Books ........................................................................................................... 66
   6.2. Papers .......................................................................................................... 66
   6.3. Online references ........................................................................................ 66
   6.4. Index of Tables ............................................................................................ 66
   6.5. Index of Figures ........................................................................................... 66
1. Introduction

In this first introduction section I want to describe the problem definitions, the goals, my used methods and of course the results of this master thesis in the methodology part.

The second part of this section introduces the necessary terminology for business process modeling.

Overview:

1.1. Methodology ......................... 10
  1.1.1. Problem definitions .............. 11
  1.1.2. Goals ................................ 12
  1.1.3. Methods ............................. 13
  1.1.4. Results .............................. 14

1.2. Definitions/Terminology ........... 16
  1.2.1. Business process ................. 16
  1.2.2. Workflow .......................... 16
  1.2.3. Model and reality ............... 17
  1.2.4. Modeling notations .............. 18
  1.2.5. Process ............................ 19
  1.2.6. Control flow ...................... 19
  1.2.7. Data flow .......................... 19
1.1. Methodology

Figure 2: Methodology of my master thesis
### 1.1.1. Problem definitions

**Issue 1: The big jumble of notations**

The practical necessity for business process modeling caused a development of multiple notations based on different sources and multiple views. Especially between computer science (CS) and information systems (IS) we find no close relation which led to a sometimes inadequate adaptation of notations for business process modeling (BPM). Computer science takes the more technical view while information systems take the more business oriented view. IS uses model types which were originally developed for software engineering for modeling business processes. This development led to a jumble and misinterpretation of graphical notations within BPM.

**Issue 2: Frequent unstructured BP modeling style and the unseen similarities between program design and BPM**

Nowadays BPM is still done in an unstructured way. This is due to the different views I described in Issue 1: Structured programming takes the more computer science oriented view; BPM the more business oriented one. The contact between these two branches is not close enough to generally recognize the similarity between program design and BPM. None of the current notations used for BPM automatically leads to structured models. Even UML activity diagrams allow the “freedom” of unstructured design.
Issue 3: Unmotivated BP decomposition on several levels of abstraction

The result of the decomposition of a business process is a strictly hierarchic structure. This means a process is split into n sub-processes (tree structure). For creating this structure no standardized approach is developed by now. The structuring of business process models is affected by the imagination and the previous knowledge of each of the model constructers. Different model designers have a different view of the real object and so they are decomposing the business processes using different criteria. Without knowing those unconscious, implicit and mostly undocumented thoughts, ideas and constraints, it is just barely possible to understand a business process model in detail.

1.1.2. Goals

Goal regarding Issue 1

To untangle the jumble of notations, a classification, description and evaluation of the most popular notations used for business process modeling is necessary. The big goal within BPM is to find a notation which combines the illustration of the control flow and data flow in one diagram in a clear and structured way. Up till now are a few notations available which combine both, but none of them is satisfying the principles of clarity and structure. I want to find ways of improvement for the most common notations to achieve this goal.

Goal regarding Issue 2 (Frequent unstructured BP modeling style)

Business process models are sometimes not transparent and so the adaptation and modification of BP models is not easy. Furthermore is the optimization of such models sometimes ineffective. The reason for those issues: BPM still happens unstructured. We can compare this situation with the 1960s, when control flow modeling happened in an unstructured manner. Software was programmed in “spaghetti code”. Source codes were complicated to read and so it was very difficult to find bugs and modify the programs. Ten years later, in the 1970s, a programming style called “structured programming” was developed to make those complicated program structures more transparent. Nowadays, all the programs are written based on the structured programming approach so the advantages are clear.

My goal is to show that we can apply the aspects of structured programming to BPM.
Goal regarding Issue 3 (Unmotivated BP decomposition on several levels of abstraction)

A diagram of a business process should be decomposable into successively finer levels of granularity on several levels of abstraction in a transparent and comprehensible way. The goal is to find a way for minimizing the complexity of decomposed BP models. Therefore we have to make it obvious why a business process model is decomposed at a specific point/position for being able to trace back this decision and for being able to completely understand the BP model.

1.1.3. Methods

Method 1 (Issue 1)

First I want to describe the historical background and development of the different graphical notations. I create a genealogical tree with the most common graphical notations to illustrate their relations by their historical development (cp. Section 3.1).

For the practical analysis of the different notations I created a single standardized example. This example is used throughout the entire thesis for showing the usability, differences and analogies of the described graphical notations. The practical example makes it easier to compare the different notations. By adapting all the described notations to this example I got a very good impression about their usability as well as their potentials for improvements, advantages and disadvantages.

Method 2 (Issue 2)

I analyze the current BPM situation with the help of my supervisor Alfred Holl. (cp. Section 2.1). Furthermore I introduce an approach of structured BPM (SBPM). In the discussion of SBPM I found big advantages and a necessity of the application of SBPM in modern BPM. In a practical example (Section 2.2) I show how an unstructured diagram can be transformed into a structured diagram.

Method 3 (Issue 3)

For finding a solution to the issue of unmotivated BP decomposition on several levels of abstraction, Alfred Holl directed my attention to the Theory of Gestalt. I discuss the Theory of Gestalt in Section 4.2 and show how it can positively affect BP decomposition in Section 4.3 of my thesis. I furthermore introduce and discuss Alfred Holl’s “feature oriented EPC” as an important approach in BPM (cp. Section 4.6).
A practical example helps me to demonstrate the positive influences of those feature oriented EPCs to BPM in Section 4.6. For achieving the aimed goals, a lot of research work was necessary. This research work is mostly based on theoretical papers, reports, books and online sources. Furthermore I got a lot of inspiration and interesting ideas from meetings and discussions with my supervisor Alfred Holl and my examiner Sara Eriksen.

1.1.4. Results

Results of Issue 1

- Genealogical tree of graphical notations
- Proposals of improvements
- Overview of analogies of the different graphical notations

By studying the historical development of graphical notations I was able to create a genealogical tree of the most common notations used for BPM. I classify the notations by their historical development (release date) and their relational interactions. Please see Section 3 for more information about this topic.

In applying my standardized example to all my described graphical notations I found potentials for improvements, which I describe in Section 3. Good diagramming notations should be an aid to clear thinking, easy to understand, designed with minimum types of symbols, easy to draw by hand and they should include structured symbolism for loops. Good diagrams help people to understand complex ideas.

Result of Issue 2

- Proof of the necessity of SBPM shown on an example
- Requirements to SBPM
- Demonstration of the defects caused by unstructured BPM style

The application of structured BPM has similar advantages to those that structured programming had for software development. Business process models become more transparent; therefore, they
are graphically and verbally documented more easily. The modification and adaptation of business process models becomes easier as well as the mapping of business process models to workflow management systems (WFMS). I describe the requirements to successful SBPM in Section 2.3. BPM has to adopt the features of structured programming to make business process models more transparent, which simplifies their graphical and verbal documentation as well as their modification and adaptation. Please find a demonstration of the defects caused by an unstructured BPM style in Section 2.2.

Result of Issue 3: Proof of logic decomposition on several levels of abstraction

For optimizing business processes, we have to split up their model representations for minimizing their complexity. In using the aspects of Theory of Gestalt we can improve the decomposition process in a positive way: Theory of Gestalt helps us to identify features in BP models which make the decomposition process more transparent and comprehensible. Model designers who were not involved in the decomposition process are now able to trace back and understand a complicated, decomposed BP model within a short period of time. Please see Section 4 for more information.

1.2. Definitions/Terminology

Before I start with the main topics of my master thesis, I want to discuss the basic terminology used for business process modeling (BPM).

1.2.1. Business process

David K. Carr, Henry J. Johansson, McGraw-Hill: ‘A process is a set of linked activities that take an input, transform it, and create an output. Ideally, the transformation that occurs in a process should add value to the input and create an output that is more useful to and effective for the recipient.’ (Carr, 1995:9)

Hammer and Champy give a similar definition: A business process is a set of activities for which one or more different inputs are required. A business process creates a result of benefit for the customers. An example is the development of a new product. The business process is lead by a responsible person who should be a part of the upper management.
The third definition I want to give was published by Scheer and Jost: A business process is an exemplary description, a function in a company in its execution and temporal dependency. Functions are activities, connected to their initiating events. Scheer does not differentiate between a ‘business process’ or a process chain. One business process extends over more functional steps.

Summarizing, I want to define a business process as follows:
A business process is a temporal-logic activity or a course of activities in an organization or organizational unit.
A business process creates benefit corresponding to the process goals.
A business process can be described from different views.
A business process can be decomposed into different levels of detail.

### 1.2.2. Workflow

Since 1993 the Workflow Management Coalition (WfMC) has been working with standards in the field of workflow management. This coalition consists of research institutes, software producers, and users. The WfMC defines a workflow as an entirely, or just partially, automated business process where documents, information, or tasks are transferred from one participant to another for execution based on procedural rules. (WfMC - Workflow Management Coalition, 2007 [online])

Gadatsch defines a workflow as a formally described, entirely or just partially automated business process. A workflow includes the temporal, functional- and resource-based specifications, which are necessary for an automatic control of an operation on the operative level. The single process steps are designated to be executed by collaborators or by an application program.’ (Gadatsch, 2005:40f, translated by KG)

### 1.2.3. Model and reality

Modeling always means abstraction, a concentration on what is important at a certain point in time. Therefore the modeling of the company reality is time dependent. (cp. Staud, 2006:2, translated by KG)
Following Staud’s definition, simplification is a necessity for models and model design. The operative reality is complex and sometimes complicated what forces the model designer to simplifications.
In his paper “Business process decomposition and Theory of Gestalt” Holl points out that “there is sometimes no distinction between real courses of activities and the models of those real courses of activities in business process modeling. A very detailed differentiation is important to avoid misunderstandings.”

I want to discuss those important differences in the following table:

<table>
<thead>
<tr>
<th>Reality (course of events in a organization)</th>
<th>Model (business process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real course of activities in a organization (e.g. processing of order 4711)</td>
<td>Business process instance (Business process instance as a model of the “processing of order 4711”)</td>
</tr>
<tr>
<td>Quantity of similar courses of activities in a organization (e.g. processing of orders 1-5000)</td>
<td>Business process (type) (Business process (type) as model of the “processing of orders 1-5000”)</td>
</tr>
</tbody>
</table>

Table 1: Model versus reality (Holl, 2000, translated by KG)

Holl furthermore discusses that we can distinguish three levels of existence in business process modeling (cp. Holl, 2000:2, translated by KG):

A real course of activities in an organization
A business process instance as a model of a real course of activities in an organization
A business process (type) as a model of a set of similar real courses of activities in an organization.

Business processes are typically modeled with the aim that parts of their functionality are supported or implemented by software. In contrast to workflow models which have the aim for an execution by a workflow management system (WFMS). “Workflow models can be derived from business process models by refinement and IT-support (cp. Gadatsch, 2005:66f, translated by KG)
1.2.4. **Modeling notations**

For modeling the reality we are in need of formal notations. Formal notations for process modeling can be distinguished in:

- Script based notations, and
- Graphical notations.

Script based notations allow a description of process models with a formal notation as we use it in programming languages. This leads to a high precision of the model specification. Unfortunately the interpretation of such models requires detailed method knowledge what makes their use and discussion in practice difficult.

I want to lay the main focus of this master thesis on the most common graphical notations in business process modeling. Please see Section 3 for detailed descriptions, illustrations and examples of the most common graphical notations used in BPM.

1.2.5. **Process**

A process describes a series of events, actions, functions, tasks and activities bringing about a result. Synonymous terms are action, activity, function and production step. Objects are processed in process steps and exchanged between them. Examples are orders, customer complaints or offers.

1.2.6. **Control flow**

The control flow shows how events, actions, functions, tasks and activities are sequenced in a business process or program. This means that the control flow illustrates the course logic, the order of the sequence in a program or business process. Those events, actions, functions, tasks or activities require completion (success, failure or completion) before moving on to the next task. Data may accompany the control but is not dominant.
1.2.7. Data flow

In contrast to the control flow, the data flow determines how data flows through a program or business process. Every data flow illustrates how data moves. Data is always executed as soon as their operands are available. In data flow modeling we have no need for program counters or sequence controls. The reasoning within data flow modeling is about data availability, transformation and latency. As data moves, control can be activated. Data can be e-mails, faxes, documents - or simply said all possible data objects. Synonymous terms to data flow are: information flow and document flow.

(cp. SQL Server-Expert, 2007 [online])
2. Structured Business Process Modeling (SBPM)

In this section I discuss the current situation - the unstructured modeling style - in business process modeling. After this general overview I demonstrate the caused defects by an unstructured BPM style. After a practical example for transforming unstructured diagrams into structured models, I summarize the requirements to structured BPM in the end of this section.

Overview:

2.1. Current situation/The unstructured modeling style in BPM .......... 21
2.2. Demonstration of the defects caused by unstructured BPM style .... 22
2.3. Requirements to structured BPM............................................. 28
2.1. Current situation/The unstructured modeling style in BPM

(Holl, 2005)

In the 1960s, control flow modeling was done in an unstructured manner, as software was programmed in so-called “spaghetti code”. Source codes were complicated to read and so it was very difficult to find bugs and modify the programs. Ten years later, in the 1970s, a programming style called “structured programming” was developed to make those complicated program structures more transparent. Nowadays, most of the programs are written based on the structured programming approach so the advantages are clear.

The application of structured BPM would have similar advantages compared to those that structured programming had for software development. Alfred Holl and Gregor Valentin point out the advantages in their paper “structured BPM” as follows:

Business process models would become more transparent; therefore, they would be graphically and verbally documented more easily.
The modification and adaptation of business process models would become easier.
The optimization of business process models (business process reengineering) would be done in a more efficient way.
The mapping of business process models to workflow management systems (WFMS) would become more effective.

Nearly 40 years after the introduction of structured programming, BPM is still done with an unstructured style. Why do still nowadays BP modelers not see the necessity of structured design?

‘This is due to the different views in computer science and information systems: structured programming takes the more computer science oriented view, BPM the more business oriented one. The contact between these two branches is not close enough to generally recognize the similarity between program design and BPM. Therefore, it has to be made explicit in detail.

Most of the program design notations and most of the procedural programming languages do not enforce structured design and coding, although they allow it. In these cases, the programmer’s effort is required. The famous exception is the very restrictive Nassi-Shneiderman diagram which only allows structured software design. […]

‘None of the current notations for BPM automatically leads to structured models. State-transition diagrams and UML activity diagrams let the ‘freedom’ of unstructured design.’ (Holl, 2005:4ff)
Alfred Holl and Gregor Valentin also point out that this statement also applies for event driven process chains (EPC). In their paper about structured business process modeling, they describe that A.-W. Scheer, the inventor of EPCs, does not give a description of the correct use of nested alternatives and loops - a specific symbol for loops is not even introduced. BPM has to adopt the features of structured programming. Alfred Holl and Gregor Valentin summarize those characteristic features as follows:

- block structures: BEGIN-END blocks, IF-ENDIF blocks, CASE-ENDCASE blocks, LOOP-ENDDO blocks instead of GOTO instructions
- hierarchically nested structures (LIFO principle: the block opened as the last one has to be closed as the first one) instead of overlapping structures
- hierarchic modular structure (vertical decomposition by subroutines)
  (cf. Holl, 2005:7)

### 2.2. Demonstration of the defects caused by unstructured BPM style

As unstructured BPM is widespread, I would like to illustrate the caused defects in an example of the current unstructured BPM style by giving an example how an unstructured BP model can be transformed into a structured BP model. The model represents a business process in a car rental company and is illustrated as a UML activity diagram. Please see Section 3.14 for further information about modeling with UML activity diagrams.
Figure 5: Typical example of the current BPM style in the form of a UML activity diagram

Explanation of the process: The start event is triggered when a customer wants to rent a car. He has particular expectations of the vehicle, e.g. he wants to be able to transport nine people. This leads to the first activity in the model, called “identify product”. The salesman has to decide which product meets the specific needs of the customer. The next activity is to check the availability of the product during the period required.

This can mean that another employee checks the car pool database to see whether the vehicle chosen is available.
After that, the first decision has to be made. If the vehicle is available (not rented by anyone else) and usable (no defects), the end of the process is reached and the bus is rented to the customer. Otherwise, if the vehicle is not available, we have to continue the process with the marked part. The intention is to make the product available for the customer. If the vehicle is not purchasable or its purchase is not useful, it could be rented from another car rental company. Otherwise, if the company considers it as useful to own a vehicle which is able to transport nine people, the company will buy it. The process ends in both cases.

Just imagine a more complex example with overlapping loops and alternatives leaving the loops in arbitrary places. You can encounter comparable situations in many business process models. This nightmare of every modern programmer is still common within current BPM without being exposed to hard criticism.

Why is the model not effective? Why it is unstructured? Let’s focus on the marked part of Figure 7 first. The positive branch of decision (3) is interrupted by decision (4). Supposing that (4) is left on its positive branch, its predecessor (3) will remain unclosed. In other words, decision (3) will not be closed in one defined point. The two decisions are overlapping, but not nested. The same situation can be found with regard to decision (1) and (2). To explain the same problem in different words, we can say this: if decision (2) is left on its positive branch, it is not closed before decision (1) is closed. This violates the LIFO principle of correctly nested alternatives as the inner alternative (2) is not completely contained in the yes branch of the outer one (1).

In the next step, the features of structured programming will be applied to the example. A transfer into a structured model follows in order to show that the principles of structured programming are applicable to BPM as well. First, the marked inner block is going to be remodeled. To nest decision (4) completely into the positive branch of decision (3), it is necessary to double the function “rent product”.

Doing so, as shown in the improved diagram (Figure 7), the inner decision (4) which was opened as the last one is closed first. It is now nested in the outer one (3).
Figure 6: Typical example of the current BPM style in the form of a UML activity diagram
Figure 7: Improved business process model in the form of a UML diagram

The entire model has not yet a structured form (decisions (1) and (2)). To develop this, the marked part is doubled like before in order to use one for the no-branch of decision (1) and the other one for the no-branch of decision (2). It is true that this would lead to a structured model, but also to a complex diagram. Because of this problem, the two decisions are combined to one. The result is shown in Figure 8. Now the model is completely structured, but it is entirely equivalent to the original example.
2. Structured business process modelling (SBPM)

Figure 8: Well structured business process model in the form of a UML diagram

This illustrates that BPM can be done in a structured way similar to structured program design.

Alfred Holl and Gregor Valentin demonstrated in their research paper that the principles of structured programming are applicable to BPM. Considerable improvements should take place with regard to the current style of BPM.
2.3. Requirements to structured BPM

(cf. Holl, 2005:14f)

Block structures should be used instead of mere control flow lines (corresponding to GOTO instructions): the notations for all of the elementary components without the mere sequence must comprise a divergent delimiter (begin) and a convergent delimiter (end, synchronization); the delimiters have to be arranged symmetrically in a diagram: IF - ENDIF (cf. BEGIN XOR - END XOR); CASE - ENDCASE; LOOP - ENDOOP; BEGIN AND - END AND; BEGIN OR - END OR.

In the case of concurrent block structures, hierarchically nested structures (LIFO principle: the block opened as the last one has to be closed as the first one) should be used instead of overlapping structures.

A motivated hierarchic modular structure (in accordance to subroutines) should be used to decompose processes vertically.

BPM notations should be urgently extended by symbols for iterations. Within structured programming, iterations have turned out to be a very important elementary component for control flow models.

‘The advantages of structured programming have been proven by an experience of 30 years. The advantages of structured BPM will be similar’ (Holl, 2005:15).
3. Graphical notations of business process models

This section is dedicated to the most common graphical BPM notations. I begin this section with an overview about the different graphical notations, illustrated with a genealogical tree. After this genealogical tree I describe every notation and apply it to a standardized example. An evaluation with strengths and weaknesses profile follows in the end of every description including ideas for improvement for some graphical notations.

Finally I present a table with the analogies of the different graphical notations.

Overview:

3.1. Genealogical tree of graphical notations ........................................... 30
3.2. Distinction of the different graphical notations ................................... 32
3.3. Control flow chart ............................................................................. 37
3.4. Network modeling technique (NMT) .................................................. 42
3.5. Petri net .............................................................................................. 47
3.6. Structure diagram (Nassi Shneidermann) .......................................... 53
3.7. SADT/IDEF0/IDEF3 .......................................................................... 57
3.8. Data flow diagrams (SA, SSA) ............................................................ 61
3.9. Swim lane diagram ............................................................................ 67
3.10. Block diagram .................................................................................... 70
3.11. Event-driven process chain ................................................................. 73
3.12. PROMET diagram ............................................................................. 77
3.13. UML activity diagram ........................................................................ 81
3.14. Extended event driven process chains (eEPC) .................................. 86
3.15. Business process diagram (BPD) ....................................................... 91
3.16. Object oriented event driven process chain (oEPC) ......................... 95
3.17. Analogy of the elementary components of control flow models and business process models .................................................. 100
3.1. Genealogical tree of graphical notations

Figure 9: Genealogical tree of diagramming notations
After the basic research work about the most used graphical notations in BPM, I discovered a lot of influences and relations of the several graphical notations. I want to visualize the genealogical dependencies with the help of a genealogical tree about graphical notations (Figure 9).

As we can clearly see was the control flow chart the first graphical notation and therefore it has a lot of successors. Control flow charts were developed in 1946/57 by Goldstine and Von Neumann. (For more information please see Section 3.4.). Nearly every following notation was influenced by control flow charts: The Nassi Shneiderman notation (1972), Block diagrams (1986), PROMET (1993), Event driven process chains (EPCs) (1992), Swim lanes (1980) and UML activity diagrams (1997). The influence to the SADT/IDEF0 notation is illustrated with a dashed arrow. This means that the notation was only weakly influenced by control flow charts.

A few years later in 1967 the first network modeling technique (NMT) was developed by Dupont de Nemours in the US. (cp. Section 3.5.). The NMT notation strongly influences Scheer’s event driven process chain (EPC) and the IDEF3 notation.

In 1962 Carl Adam Petri developed the Petri net notation, our third “father” notation according to the genealogical tree of graphical notations. Petri nets have a strong influence to the data flow diagram notation (DFD/SA/SSA) and to UML activity diagrams. An indirect influence can be noted in studying the concepts of the EPC notation.

Regarding to the genealogical tree of graphical notations we can see that every graphical notation, developed after 1962 represents a successor of our three “father notations”: Control flow charts, the network modeling technique and Petri nets.

The newest notation recorded in my genealogical tree are BPDs (Business process diagrams), an advancement of the PROMET notation (Section 3.13).

The UML activity diagram is the only graphical notation within the UML family which is considered as useful for BPM.

Because of the fact that the IDEF3 notation is widespread and the distinction between IDEF0 and IDEF3 is sometimes unclear, I included IDEF3 (cp. Section 3.8.4). The IDEF3 notation is a successor of the SADT/IDEF0 notation, but its purpose is the modeling of workflows and not business processes. That is the reason why the IDEF3 notation is illustrated in bracelets in the genealogical tree of graphical notations.
3.2. Distinction of the different graphical notations

Basically, we can distinguish between two types of graphical notations in BPM:
Data flow oriented,
Control flow oriented notations, and
Data flow and control flow oriented notations.

Table 2 shows which graphical notation supports the control-flow aspect, the data-flow aspect or both, also regarding the classical notations as well as the object oriented ones.

<table>
<thead>
<tr>
<th>Model type</th>
<th>Data flow oriented</th>
<th>Data- and control flow oriented</th>
<th>Control flow oriented</th>
</tr>
</thead>
</table>
| Classical notations             | Data flow diagram (SA, SSA), Section 3.9                                           | SADT/IDEFO, Section 3.8 PROMET (Process Method) diagram, Section 3.13 Business process diagram, Section 3.16 | Control flow chart, Section 3.4  
Network modeling technique, Section 3.5  
Petri net, Section 3.6  
Structure diagram (Nassi-Shneiderman), Section 3.7  
Swim lane diagram, Section 3.10  
Block diagram, Section 3.11  
Event driven process chain (EPC), Section 3.12  
Extended event driven process chain (eEPC), chapter 3.15 |
| Object oriented notations       | e.g. UML-Use case diagram (only minor usable for BPM)                              | Not available                                    | UML - Activity diagram, Section 3.14  
Object oriented event driven process chain (oEPC), Section 3.17 |

Table 2: Different model types and notations
It is important to mention that the notations from the column “data flow oriented” are only applicable for business process modeling if there is a way of representing the control flow aspect in the notation.

### 3.2.1. Object orientation in BPM

Until the 1990s, software design was dominated by the paradigm of so-called procedural software design. This paradigm very clearly distinguishes between data and the procedures (operations) that process the data. In the 1970s, a new paradigm was created, the so-called object-oriented software engineering which established itself in practice in the 1990s (cp. Oesterreich, 2004:12f, translated by KG).

The object oriented approach is a very interesting one because it leads us to the point where a separation between a dynamic and a static aspect of an application area is partly removed. For business process modeling, dynamic is of particular importance because business processes are mostly representing dynamic aspects in a company (Staud, 2006:310, translated by KG).

### 3.2.2. UML and business process modeling

UML offers a flexible and extendable notation. In its origin, it was created for the modeling of software and did not include all elements which are necessary for the modeling of business processes. According to defined formal rules, enhancements can be made (so-called profiles).

UML is standardized by the OMG (Object Management Group) and so it is an industrial standard. Every famous company is supporting UML or is involved in its further development: IBM, Microsoft, Sun, Hewlett-Packard, or Oracle for example.

The formal foundation of UML is stronger than, for instance, with event driven process chains. Furthermore there is a big formal analogy in modeling hard- and software systems and in modeling businesses. (Oesterreich, 2004:13 und 147, translated by KG)

For modeling business processes UML activity diagrams play an important roll (cp. Section 3.14 for more details). All the other UML notations are just minor relevant for BPM.

For the illustrations and descriptions in this master thesis, I use UML in Version 2.0.
3. Graphical notations for business process models

3.2.3. Standardized Example

For illustrating the different notations, for showing their usability and for enabling their comparison I created an example, which includes loops, conditions and parallelism. Every graphical notation will be illustrated with the help of this example in the end of their particular description sections in an unstructured and structured way.

The example I created, represents a simplified order process in a trading company.
Explanation of the process:

The company receives an order from a customer. After recording the order, the order has to be checked. If the ordered product is not available on stock, the company has to order the required product. After this step, an order confirmation is sent to the customer.

As soon as the required goods are available, the delivery process will be initiated. During this shipping process, the invoice will be sent to the customer. 14 days after the delivery, the payment has to be checked. If the payment is not recorded on the company’s account, a payment reminder will be sent to the customer. After another 2 weeks the payment has to be checked again. If there is still no payment recorded, another payment reminder will be sent to the customer. After a second 2 weeks period, the payment will be checked again and other actions have to be taken to force a payment (this is not part of this process). As soon as the payment is recorded, the order will be closed.

I illustrate this example process in an UML activity diagram. For a better distinction between control flow and data flow, I use my improved notation, which I describe in Section 3.14: The dashed arrows illustrate the data flow of the process.

3.2.4. Evaluation of the graphical notations

After applying my standardized example to the different graphical notations I got a very good overview over the strengths and weaknesses of every notation. I decided to create a strengths and weaknesses profile for every one of the mentioned graphical notations with the help of the following categories and their evaluation schemes:

1.) Clarity and clearness

How good is the diagram readable? How long does it take to figure out its purpose? Good diagrams help people to understand complex ideas. A diagramming notation should be designed to help thinking and communication and for computer-aided thinking (cp. Martin, 1987:56ff).

10 points: best result

2.) Easy to draw by hand?

Do I need a computer or a special program to draw the diagram or is the notations symbolism so trivial that I can draw it by hand? The use of decoration that does not enhance visual logic should be avoided. The instinct to be artistic must be suppressed and replaced with an urge to maximize clarity (cp. Martin, 1987:56ff).

10 points: best result
3. Graphical notations for business process models

3.) Easy to understand?
How long does it take to understand the diagram in detail, especially without former knowledge about the notation? The diagrams should use constructs that are obvious in meaning and as familiar as possible (cp. Martin, 1987:56ff).

10 points: best result, the diagram is very easy to understand, even without former knowledge.

4.) Required space
Does the diagram need a lot of space?
10 points: best result, the diagrams required space is adequate.

5.) Structured Design mandatory?
Is structured design mandatory for this graphical notation?
0 points: Structured design is not possible.
5 points: The graphical notation is not forcing the model designer to a structured design, but the notation allows it.
10 points: Structured design is mandatory.

6.) Structured symbolism for loops included?
Is there a possibility for the structured illustration of loops?
0 points: No, the illustration of structured loops is not included in the notation.
10 points: Yes, structured symbolism for the illustration of loops is included in the notation.

7.) Symbolism for decisions included?
Is there a possibility for the illustration of decisions?
0 points: No, the illustration of decisions is not possible.
5 points: No, but there is an indirect way for the illustration of decisions.
10 points: Yes, symbolism for the illustration of decisions is included in the notation.

8.) Parallelism supported?
Is there a possibility for the illustration of parallelism?
0 points: No, the illustration of parallelism is not possible.
5 points: No, but there is an indirect way for the illustration of parallelism.
10 points: Yes, symbolism for the illustration of parallelism is included in the notation.
9.) Data flow aspect supported?
Does the graphical notation support the data flow aspect?
0 points: No, the data flow aspect is not supported by the original definition.
5 points: No, the data flow aspect is not supported by the original definition, but there is an improvement possibility.
10 points: Yes, the data flow aspect is supported by the original definition.

10.) Control flow aspect supported?
Does the graphical notation support the control flow aspect?
0 points: No, the control flow aspect is not supported by the original definition.
5 points: No, the control flow aspect is not supported by the original definition, but there is an improvement possibility.
10 points: Yes, the control flow aspect is supported by the original definition.

The results of this evaluation are found after the demonstration of every described graphical notation in Section 3 in form of a short description and a strengths and weaknesses profile.

### 3.3. Control flow chart

The first recorded systematic development of the flow chart notations was recorded 1946 by Goldstine and Von Neumann. In the 1970s, Gane and Sarson popularized the notation for business modeling.

In the basic flow chart notation, three different symbols are used: Squares for activities and tasks, diamonds for decision points, and arrows for representing the control flow.

It’s important to note the difference between flow charts and data flow diagrams (cp. Section 3.9). Unlike data flow diagrams, control flow diagrams describe the control flow and the detailed logic of a business process or business rule (cp. Hebb, 2006, [online]).
## 3.3.1. Notation of control flow charts

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Terminator" /></td>
<td>Terminator, Start or End</td>
<td>Shows the materials, information, or action (inputs) that start the process, and the results (output) at the end of the process.</td>
</tr>
<tr>
<td><img src="image" alt="Data" /></td>
<td>Data</td>
<td>In- and output of data</td>
</tr>
<tr>
<td><img src="image" alt="Process" /></td>
<td>Process</td>
<td>Illustration of a process activity</td>
</tr>
<tr>
<td><img src="image" alt="Alternative" /></td>
<td>Alternative</td>
<td>Illustration of alternatives or decisions. Normally the control flow is split up in two directions (true/false)</td>
</tr>
<tr>
<td><img src="image" alt="Control flow" /></td>
<td>Control flow</td>
<td>Connection to the other elements</td>
</tr>
<tr>
<td><img src="image" alt="Sub-course/Sub-program" /></td>
<td>Sub-course/Sub-program</td>
<td>Link to other flow charts.</td>
</tr>
<tr>
<td><img src="image" alt="Connector" /></td>
<td>Connector</td>
<td>Connectors are used for combinations after alternatives or for reasons of clearness to prevent a crossing of lines.</td>
</tr>
<tr>
<td><img src="image" alt="Iteration/Loop" /></td>
<td>Iteration/Loop</td>
<td>The iteration defined in the upper square is executed until a certain condition becomes true.</td>
</tr>
</tbody>
</table>

*Table 3: Control Flow Chart (DIN66001, 1966)*
3. Graphical notations for business process models

3.3.2. Demonstration of a flow chart diagram

Figure 11: Demonstration of a flowchart diagram – Original notation
3.3.3. Evaluation of the control flow chart notation

Control flow charts are very easy to draw by hand and easy to understand. The notation offers symbols for decisions and loops. Parallel activities can be illustrated with the help of a while-function because a direct symbol for parallelism is not included in the notation. Process documents are illustrated with a special symbol. Just the arrows for control- and data flow are the same. For optimizing the notation, a distinction between control- and data-flow would be appreciated (cp. Section 3.4.5).

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>10</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbol for loops included?</td>
<td>10</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>5</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

Figure 12: Strength and weaknesses profile of control flow charts

3.3.4. Idea for improvement of control flow diagrams

As mentioned in Section 3.4.3, a distinction between the symbolism for control and data flow would be appreciated in the control flow chart notation.

The easiest way for a distinction would be the use of dashed arrows for the data flow.

Please see Figure 13 for a demonstration of this idea for improvement.
3. Graphical notations for business process models

---

**Figure 13: Demonstration of a control flow chart - Improved notation**

- Order received
- Record order
- Check order
- Product on stock? (yes/no)
- Order product
- Create order confirmation
- Send order confirmation
- Create invoice
- Ship goods
- Send invoice
- Check payment
- WHILE payment not recorded
- Create payment reminder
- Send payment reminder
- Check payment
- Close order
- Order closed

---
3.4. **Network modeling technique (NMT)**

The first methodology for the network modeling technique (NMT) was developed in 1957 by Dupont de Nemours in the US and is called ‘CPM’ (Critical path method). ‘PERT’ (Program Evaluation and Review Technique) was developed for the construction of the Polaris rocket in 1958. At the same time, ‘MPM’ (Metra Potential Method) for reactor construction was developed by METRA. All of those methods are based on one graphical model representation which is called network modeling technique. (cp. Heinrich, 2003:214, translated by KG)

‘In a project, an activity is a task that must be performed and an event is a milestone marking the completion of one or more activities. Before an activity can begin, all of its predecessor activities must be completed. Project network models represent activities and milestones by arcs and nodes.’ (Internet Center for Management and Business Administration, 2007 [online])

3 different types of networks can be distinguished:
- event on node networks (cf. PERT)
- activity on node networks (cf. MPM)
- activity on arc network (cf. CPM)

### 3.4.1. Event on node network

Activities are represented by arrows and events by nodes. The primary interest lies in the events. Event on node networks enable probabilities calculations. PERT uses the event on node network and is based on a relatively complicated mathematical concept, which allows a trivalent appraisal.

### 3.4.2. Activity on node network

Activities are represented by nodes (normally rectangles with rounded corners); events are not illustrated. In contrast to the ‘activity on arc’ or ‘event on node’ network, an activity can start, when the predecessors are only partially finished. A certain level of completion of the predecessors is enough. Arrows are illustrating the course of the activities.

Heinrich (2003:220) points out three reasons why ‘activity on node’ networks are more advantageous then the other NMT notations in business process modeling:
- Nodes can contain all important information concerning the activity, without influencing the clarity of the diagram in a negative way.
Pseudo activities - except start and end - are not necessary. The clarity is not influenced by pseudo activities.

Changes in the structure of the model are easily applicable. Arrows can easily be added or removed without the need to re-calculate the model.

### 3.4.3. Activity on arc network

Every activity (represented by an arrow) belongs to one start- and one end-event (represented by circles). This requires that every activity has to be terminated before another activity can start. Except for the start- and end-event, every node represents a start- and an end-event for different activities. An event can be start- or end-event for more than one activity. However: two events can just be connected by one activity. That is why we need “pseudo activities” for illustrating the parallelism of activities. The use of pseudo activities is also necessary when two or more activities with different start- and end-events are connected with each other. A pseudo activity is represented by a dashed arrow. Please see Figure 15 for an illustration of a pseudo activity. (cf. Heinrich, 2003:218, translated by KG)

### 3.4.4. Activity Table

Before we can begin with the network model, it is necessary to list the different activities in an activity table. The following table shows the activities for the introduced example of an order process:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Activity label</th>
<th>Predecessor</th>
<th>Required time (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Record and check order</td>
<td>--</td>
<td>0,5</td>
</tr>
<tr>
<td>B</td>
<td>Order product if necessary</td>
<td>A</td>
<td>0,2</td>
</tr>
<tr>
<td>C</td>
<td>Create order confirmation</td>
<td>A, B</td>
<td>0,1</td>
</tr>
<tr>
<td>D</td>
<td>Send order confirmation</td>
<td>C</td>
<td>0,1</td>
</tr>
<tr>
<td>E</td>
<td>Create invoice</td>
<td>D</td>
<td>0,1</td>
</tr>
<tr>
<td>F</td>
<td>Send invoice</td>
<td>E</td>
<td>0,1</td>
</tr>
<tr>
<td>G</td>
<td>Ship order</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>Check payment</td>
<td>E</td>
<td>0,2</td>
</tr>
<tr>
<td>I</td>
<td>Create payment reminder</td>
<td>H</td>
<td>0,1</td>
</tr>
<tr>
<td>J</td>
<td>Send payment reminder</td>
<td>I</td>
<td>0,1</td>
</tr>
<tr>
<td>K</td>
<td>Close order</td>
<td>J</td>
<td>0,2</td>
</tr>
</tbody>
</table>

Table 4: Table of activities
3.4.5. **Time analysis**

The following questions have to be answered:

How much time does the process/the project need at least? (generally the shortest time in which the activity can be completed? (optimistic time))

Which activities are influencing the optimistic time and what are their latest possible start dates, so that the minimal or optimistic time can be reached?

Which activities can be postponed or enlarged without influencing the minimal duration? In other words: Where are the buffer times and how big are they?

For answering those questions is it necessary to find out the activity durations. This happens by estimation or measuring. PERT uses three different activity durations, for example:

- **Optimistic time:** generally the shortest time in which the activity can be completed.
- **Most likely time:** the completion time having the highest probability.
- **Pessimistic time:** the longest time that an activity might require.

The expected time for each activity can be approximated using the following weighted average:

\[
\text{Expected time} = \frac{(\text{Optimistic time} + 4 \times \text{Most likely time} + \text{Pessimistic time})}{6}
\]

The results of the measurement or estimation are collected in the Table of activities and later illustrated in Figure 15.

**3.4.5.1. Notation of an activity-on-arc network:**

A...Number or label of the activity, based on the activity table
1...Duration of the activity in time units (days, weeks, months)
2...Estimated requirement of capacity in units of the requested resource
3...Event number
4...Earliest start date of the activity in time units after the start date
5...Latest start date of the activity in time units after the start date

![Figure 14: Notation of an activity on arc network (Heinrich, 2003:222)](image-url)
The critical path, which runs without interruption from the start activity to the end activity, is represented by a thick black line. The critical path represents the critical activities where the time buffer is null.

### 3.4.6. Demonstration of an activity on arc network

The following demonstration shows the network model of the order example, based on the activity table with eleven activities (A to K); all important relationships are easily cognizable. The earliest possible end date of activity K, and so for the entire process, is in time unit 1,3. The latest end date is in time unit 29,3 (planned end date). The model contains four pseudo activities (illustrated by dashed arrows).

![Diagram](image)

Figure 15: Demonstration of an activity on arc network

### 3.4.7. Evaluation of the NMT activity on arc networks notation

NMT activity on arc networks are not understandable without the corresponding activity table. The notation offers no possibility for a direct naming of the different activities in the diagram. Another problem talking about clarity and clearness are the pseudo activities. Pseudo activities are necessary for illustrating the process logic in our order example, but they are making a structured design impossible. The data flow aspect is not regarded, but NMT activity on arc networks are a good tool for time planning and analysis of business processes.
3.4.8. Important difference to other graphical notations

In one of our meetings concerning this master thesis, my supervisor Alfred Holl and I decided to point out an important difference of the network modeling technique (NMT) and Petri Nets (cp. Section 3.6), as compared to the other notations.

In Section 3.5, I presented the three different types of NMT notations:

- Activity on arc networks,
- Activity on node networks, and
- Events on node networks.

When a business process should be represented with an ‘activity on arc network’, the activities are represented by arrows, as indicated by the name. The nodes represent the different states of the process:

![Figure 17: Schema for activity on arc networks](image)
In the Petri net notation (cp. Section 3.6. for further information) and in the UML transition diagram notation, we can also find nodes as representations of these various states. As in the previous example, the activities are represented by arrows.

All the other graphical notations which are more relevant for BPM like flow charts, EPCs, Nassi-Shneiderman-diagrams, etc., represent activities by nodes, as follows:

![Figure 18: Schema for activity on node networks](image)

### 3.5. Petri net

‘Petri net theory allows a system to be modeled by a Petri net, a mathematical representation of the system. Analysis of the Petri net can then, hopefully, reveal important information about the structure and dynamic behavior of the modeled system. This information can then be used to evaluate the modeled system and suggest improvements or changes. Thus, the development of a theory of Petri nets is based on the application of Petri nets in the modeling and design of systems.’ (Peterson, 1981:1)

In 1962, Carl Adam Petri gave the initial impulse for the development of Petri nets in his dissertation “Kommunikation mit Automaten” [Communication with automata] supported by the Massachusetts Institute of Technology (MIT, Boston/USA).

Petri Nets are designed specifically to model systems with interacting concurrent components (cp. Heinrich, 2003:380ff, translated by KG).


Places (Circles) which represent the static states of processes (documents, data, resources, etc.). Those passive system components can be embedded or stored. Places are also called states or channels. For representing the dynamic behavior of the net, places can contain marks. (cp. Figure 18)

Transitions (rectangles) represent the transition of information (functions, processes, activities). Those active system components are able to produce, transport, or change something. Transitions are also called instances or events.
Edges connect the different kinds of nodes and represent the control flow. A directed edge (an arrow) is not a system component; it’s a relation between system components. It describes, for example the coherence between system components, the access right on a system component, or the direct linking of system components.

Causal-logical coherence: Every state is initiated or aborted by at least one event; this causal-logical coherence is illustrated by directed edges (arrows).

Sometimes the elementary notation of Petri nets does not cover all requirements and so, based on the elementary notation, many so-called “advanced Petri nets” were developed.

The following table illustrates the main aspects of the most common Petri nets:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Place</th>
<th>Transition</th>
<th>Edge</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Petri-net</td>
<td>Channel/Instance-Net (C/I-Net)</td>
<td>Channel</td>
<td>Instance</td>
<td>Unweighted</td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>Condition/Event-Net (C/E-Net)</td>
<td>Condition (one mark)</td>
<td>Event (one mark)</td>
<td>Unweighted</td>
<td>Identical information content -&gt; black mark</td>
</tr>
<tr>
<td></td>
<td>Place/Transition-Net (P/T-Net)</td>
<td>Place (can contain several marks)</td>
<td>Transition (several marks)</td>
<td>Weighted with integers</td>
<td>Identical information content -&gt; black mark</td>
</tr>
<tr>
<td>Advanced Petri-nets</td>
<td>Predicate/Transition-Net (Pr/T-Net)</td>
<td>Predicate (can contain several marks)</td>
<td>Transition (can contain several marks)</td>
<td>Weighted with integers</td>
<td>Different information content -&gt; colored marks¹</td>
</tr>
</tbody>
</table>

Table 5: Elements of Structure Diagrams (Gadatsch, 2005:80, translated by KG)

In this master thesis I would like to give a short overview over P/T-nets (Place/Transition-nets) where places can contain several marks. Additionally the edges can contain quantifiers, which can transport one or more marks between the places.

¹ Predicate/Transition-nets (Pr/T nets) differ by the art of the marks. Simple Petri nets don’t distinguish between different marks (“black marks”); the marks of predicate/transaction-nets are called “colored marks”, meaning that the marks can contain information. That’s why those nets are called colored Petri nets.
### 3.5.1. Notation of Place/Transition-Nets

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| ![Caption](image) | Place | Representation of static states of processes (documents, data, resources, etc.)  
C: capacity of a place, equivalent to the amount of marks which can be held. Not applicable = endless capacity.  
● mark illustrates the current dynamic state of the control flow |
| ![Caption](image) | Transition | Representation of dynamic transitions of information (functions, processes, activities, etc.)  
C: capacity of a transition, equivalent to the amount of marks which are taken from the input places during a switching operation. |
| ![Caption](image) | Edge | Connection between places and transitions.  
W: weight of an edge, equivalent to the amount of marks which are transmitted from one place to a transition or the other way round. |

Table 6: Elements of Structure Diagrams (Gadatsch, 2005:80, translated by KG)

Figure 19 shows an example for a Place/Transition net. Before the switching operation of the transition T₁, the input places P₁ contain 3 marks. The input edge to T₁ transports 2 marks for every switching operation. The output edges also transport two marks to the places P₂ and P₃ for every switching operation. P₂ and P₃ have enough free capacity for this operation. After the switching operation, the place P₂ contains its maximal capacity. The input place P₁ has no more marks for another switching operation.

![Figure 19: Demonstration of switching operations in a P/T-Net (Gadatsch, 2005:79)](image)

The illustration shows a Place/Transition net. The coloring of marks was not applied for keeping a clear layout.
3.5.2. Demonstration of a Petri net

Figure 20: Demonstration of a Petri net (P/T) without marks - Original notation
3.5.3. **Evaluation of the P/T Petri nets without marks notation**

‘From the point of view of business process modeling Petri nets are normally seen as too complex and hard to understand. The high amount of developed variants led to inconsistent notations and so Petri nets are not common in business process modeling, especially in the field of as-is analysis and target-state analysis.’ (Gadatsch, 2005:80f)

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>Easy to draw by hand</th>
<th>Easy to understand</th>
<th>Required space</th>
<th>Structured Design mandatory?</th>
<th>Structured symbolism for loops included?</th>
<th>Symbolism for conditions included?</th>
<th>Parallelism supported?</th>
<th>Data flow aspect supported?</th>
<th>Control flow aspect supported?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarity and clearness</td>
<td>Easy to draw by hand</td>
<td>Easy to understand</td>
<td>Required space</td>
<td>Structured Design mandatory?</td>
<td>Structured symbolism for loops included?</td>
<td>Symbolism for conditions included?</td>
<td>Parallelism supported?</td>
<td>Data flow aspect supported?</td>
<td>Control flow aspect supported?</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

**Figure 21: Strengths and weaknesses profile of Petri Nets**

3.5.4. **Idea for improvement of Petri Nets**

Structured symbolism for loops:

With the use of $[]^*$ the unstructured diagram (Figure 20) can be easily transformed into a structured diagram (Figure 22). The original notation includes no structured symbolism for loops.
Figure 22: Demonstration of a Petri net (P/T) without marks - Improved notation
3.6. Structure diagram (Nassi Shneidermann)

Nassi Shneiderman diagrams are also called “structure diagrams”. Those diagrams are normally used for an abstract description of algorithms, but they can also be used in business process modeling. The diagrams describe procedural program sequences. The regarding processes are described on a code independent level (without a direct connection to a programming language) and so they are representing “top-down” design. (Grotehusmann, [online], translated by KG)

When structured programming became popular in the 1970s, Nassi-Shneiderman diagrams were introduced as tools for the design of structured program courses.

‘Nassi Shneiderman diagrams are often better organized, structured and also more understandable than other notations. That is the reason why this kind of notation is preferred for representing process specifications. But a considerable amount of diagrams is though necessary.’ (Yourdon, 1992:263, translated by KG)

The Nassi Shneiderman notation is the only graphical notation which forces the model designer to a structured design. (cp. Section 2)

3.6.1. Notation of Nassi Shneiderman structure diagrams

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Sequence" /></td>
<td>Sequence</td>
<td>The instructions of a program or business process are placed in rectangles. Within those rectangles a description of the instruction, or the instruction itself is listed. The instructions are serialized in the following order: Instruction I₁, Instruction I₂, Instruction I₃,…</td>
</tr>
<tr>
<td><img src="image" alt="Alternative (IF/THEN/ELSE)" /></td>
<td>Alternative (IF/THEN/ELSE)</td>
<td>Within a sequence is it sometimes necessary to execute one or more instructions under a certain condition. We can also call those IF/THEN/ELSE selections. Interpretation: If the condition C is true, instruction I₁ is going to be executed. Else instruction I₂ is going to be executed.</td>
</tr>
</tbody>
</table>
### 3. Graphical notations for business process models

<table>
<thead>
<tr>
<th>Case differentiation (CASE)</th>
<th>With the help of case differentiations, an instruction can have multiple exits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPEAT UNTIL Iteration</td>
<td>Iteration I is executed until condition C becomes true.</td>
</tr>
<tr>
<td>FOR Iteration</td>
<td>FOR iterations are executed for a known, certain amount of runs. An internal control variable (x) runs with an interval (a, b). As long as the control variable lies within the interval (a, b), iteration I is executed and the control variable increases (x + increment parameter). If there is no parameter listed, the increment is automatically 1.</td>
</tr>
<tr>
<td>WHILE Iteration</td>
<td>As long as condition C is true, iteration I is executed.</td>
</tr>
</tbody>
</table>

**Table 7: Elements of Structure Diagrams (Kubicek and Kamin, 2005, [online] translated by KG)**
3.6.2. Demonstration of a Structure Diagram

![Diagram of a Nassi Shneiderman diagram]

Figure 23: Demonstration of a Nassi Shneiderman diagram - Original notation

3.6.3. Evaluation of the Nassi Shneiderman notation

Nassi Shneiderman diagrams are very easy to understand - they are following the aspects of structured design. Please see Section 2 for the advantages of a structured design in BPM. Furthermore are Nassi Shneiderman diagrams easy to draw by hand. The notation includes symbolism for the illustration of for...while functions and case conditions. The only problem in modeling my introduced order example is parallelism. The Nassi Shneiderman notation allows in its original definition no parallelism. Please see Section 3.7.4. for an idea for improvement for this problem.
3. Graphical notations for business process models

<table>
<thead>
<tr>
<th>Clarity and clearrness</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>10</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>10</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>5</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>0</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 24: Strengths and weaknesses profile of Nassi Shneiderman diagrams

3.6.4. Idea for improvement of Nassi Shneiderman diagrams

An idea for improvement of the Nassi Shneiderman notation is a possibility for illustrating parallel activities. As you can see in figure 25 I split up the “Ship goods” box to illustrate the parallelism to “Create invoice” and “Send invoice”.

Figure 25: Demonstration of a Nassi Shneiderman diagram - Improved notation
3.7. SADT/IDEF0/IDEF3

SADT (Structured Analysis Design Technique) was developed by SofTech 1974/75 and is a mostly graphical description and design method, which supports a hierarchic structured system draft that is documented with diagrams. Every diagram is a finite, directed, node- and edge-marked graph. In SADT diagrams, the main focus lies on the illustration of activities (if an activity model is used) or on the illustration of the dataflow (if a data model is used). Next to the definition of the graphical description instruments, a methodic approach for illustrating the diagrams (top-down design) is predetermined. The result of the modeling is a structured decomposition of the illustrated system as a hierarchically structured quantity of SADT diagrams. (Heinrich, 2003:360f,
translated by KG)

3.7.1. SADT and IDEF0

‘IDEF0 is a notation, designed to model the decisions, actions, and activities of an organization or system. IDEF0 was derived from a well-established graphical language, the Structured Analysis and Design Technique (SADT). The United States Air Force commissioned the developers of SADT to develop a function modeling notation for analyzing and communicating the functional perspective of a system.’ (KBSI - Knowledge Based Systems Inc., 2006, [Online])

IDEF0 is a commercial tool, based on SADT.

3.7.2. SADT/IDEF0 notation

A SADT diagram consists of several activities or data models. In every model of those constructs, the smallest cohesive entity of the SADT diagram is the so-called SADT-Object. Every SADT-Object is characterized by three input arrows and one (on the right side) output arrow. The amount of SADT-Objects is limited for reasons of understanding and readability (normally ca. three to five). Bigger systems are illustrated by several SADT diagrams which are linked distinctly by connectors.

Typical for SADT is that every side of an activity and a data model has a special meaning. The left side is designated for Inputs (I), the upper side is designated for controls (C), the right side is designated for outputs (O) and the lower side is designated for mechanisms (M). This ICOM notation represents an important system principle: Inputs are transformed to outputs, controls restrict or enable transformations under certain conditions, and mechanisms describe the type of the support in a transformation (e.g. which tools, which personnel, which memory etc. is used) (cp. KBSI - Knowledge Based Systems Inc., 2006, [Online]).
### Table 8: Notation IDEF0 diagram (Gadatsch, 2005:70, translated by KG)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Action title" /></td>
<td>Action box</td>
<td>Description of an action (activity, course or process), # = node number</td>
</tr>
<tr>
<td><img src="image" alt="Arrow from left leading to the box" /></td>
<td>Arrow from left leading to the box</td>
<td>Required input for an action. Description of the input object which is transformed by the action to an output object.</td>
</tr>
<tr>
<td><img src="image" alt="Arrow from right leading away from the box" /></td>
<td>Arrow from right leading away from the box</td>
<td>Output object of an action</td>
</tr>
<tr>
<td><img src="image" alt="Arrow from the top leading to the box" /></td>
<td>Arrow from the top leading to the box</td>
<td>Predetermined object. The specifications for the execution of the action which won’t be changed by the action (e.g. guidelines, plans or laws).</td>
</tr>
<tr>
<td><img src="image" alt="Arrow from the bottom leading to the box" /></td>
<td>Arrow from the bottom leading to the box</td>
<td>Operating resources, tools, and personnel for the execution of an action. It emanates without changes from the action.</td>
</tr>
</tbody>
</table>

‘The “box and arrow” graphics of an IDEF0 diagram show the function as a box and the interfaces to or from the function as arrows entering or leaving the box. To express functions, boxes operate simultaneously with other boxes, with the interface arrows “constraining” when and how operations are triggered and controlled. The basic syntax for an IDEF0 model is shown in the figure below.’ (KBSI - Knowledge Based Systems Inc., 2006, [Online])

![Figure 26: IDEF0 box and arrow graphics (KBSI - Knowledge Based Systems Inc., 2006, [Online])](image)
IDEF0 process models are refined step by step based on detailed process illustrations. The construct of the reference object (vertical arrow leading away from the box) is used as a reference on the refined process. (cp. Gadatsch, 2005:69, translated by KG)

### 3.7.3. Demonstration of an IDEF0-Diagram

![Figure 27: Demonstration of an IDEF0 diagram](image)

### 3.7.4. IDEF3

I also want to mention IDEF3 diagrams in this master thesis because they are widespread and the distinction between IDEF0 and IDEF3 is a very important one: IDEF3 is an advancement of IDEF0 for workflow modeling. The advancements are related to the precise modeling of the control flow. IDEF3 uses its own symbols for the illustration of activities and of the control flow to a large extent but IDEF3 can reference with-in IDEF0 modeled objects. This means that IDEF3 reverts to the IDEF0 repository. Through this, a strict separation between the business process model (IDEF0) and the workflow model (IDEF3) occurs. The actions are specified in a repository. An important element is the connection box which specifies the essential conditions for the connection of the actions. (cp. Gadatsch, 2005:71, translated by KG)
3.7.4.1. **IDEF3 notation**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Action title" /></td>
<td>Action box</td>
<td>Description of an action (activity, course or process)</td>
</tr>
<tr>
<td><img src="image" alt="Link" /></td>
<td>Link</td>
<td>Control flow between actions or connection boxes</td>
</tr>
<tr>
<td><img src="image" alt="Relational link" /></td>
<td>Relational link</td>
<td>Relational dependency between two action boxes</td>
</tr>
<tr>
<td><img src="image" alt="Asynchronous connection box" /></td>
<td>Asynchronous connection box</td>
<td>Logical connection operator for asynchronous control flow (concurrent, time independent activities)</td>
</tr>
<tr>
<td><img src="image" alt="Synchronous connection box" /></td>
<td>Synchronous connection box</td>
<td>Logical connection operator for synchronous control flow (concurrent, simultaneous running activities). Connection types: AND/OR/XOR</td>
</tr>
</tbody>
</table>

Table 9: Notation IDEF3 diagram (Gadatsch, 2005:70f, translated by KG)

### 3.7.4.2. Demonstration of an IDEF3 diagram

![Demonstration of an IDEF3 diagram](image)

Figure 28: Demonstration of an IDEF3 diagram (Gadatsch, 2005:72, translated by KG)
IDEF diagrams are offering a notation for business process and workflow modeling. Since the symbolism changes at the transition from business process to workflow modeling, a break in the illustration is caused. (cp. Gadatsch, 2005:72, translated by KG)

### 3.7.5. Evaluation of the IDEF0/SADT notation

SADT/IDEF0 diagrams are easy to draw by hand. The for SADT/IDEF0 typical box and arrow approach is easy to understand. The notation offers no symbolism for loops and conditions are modeled as a control input. Parallelism is not supported either. Because of the special character of the notation, a structured design is not possible.

<table>
<thead>
<tr>
<th>Clarity and cleanness</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>7</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>0</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>5</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>0</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>0</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clarity and cleanness</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>5</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>10</td>
</tr>
<tr>
<td>Required space</td>
<td>7</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>0</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>5</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>0</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>0</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

**Figure 29: Strengths and weaknesses profile of IDEF0/SADT diagrams**

### 3.8. Data flow diagrams (SA, SSA)

‘A SSA (Structured Systems Analysis) or also called SA (Structured Analysis) data flow diagram is a graphic system model. The main focus lies on the process structure seen from the data view of a system. It concentrates on the data view, not on processes. The terms SA (Structured Analysis) and SSA (Structured Systems Analysis), next to the term “data flow diagram”, became widely accepted.’ (Heinrich, 2003:370, translated by KG)
Data flow Diagrams show no flow of control which represents the difference to flowcharts (cp. Section 3.4). ‘The data flow diagram portrays a situation from the point of view of the data, while a flowchart portrays it from the point of view of those who act upon the data. For this reason, you almost never see a loop in a data flow diagram. A loop is something that the data is unaware of; each datum typically goes through it once, and so from its point of view it is not a loop at all. Loops and decisions are control considerations and do not appear in data flow diagrams.’ (DeMarco, 1979:40)

‘In classical analysis, we first try to see operations from the user’s viewpoint; i.e. we interview him and try to learn from him how things work. Then we spend the rest of our time trying to document the working of modified operations from the system’s viewpoint. (Notice that this approach is pervasive in unstructured technology: a flowchart, for instance, is design documentation from the system’s point of view’. (DeMarco, 1979:49)

### 3.8.1. Notation of data flow diagrams

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Data flow" /></td>
<td>Data flow</td>
<td>A data flow is a pipeline through which packets of information of known composition flow. A data flow is <em>not</em> a representation of flow of control, the stream of consciousness of the computer or person that processes the data!</td>
</tr>
<tr>
<td><img src="image" alt="Process" /></td>
<td>Process</td>
<td>Processes invariably show some amount of work performed on data. An activity is a transformation of incoming data flow(s) into outgoing data flow(s). In a completed set of Data Flow Diagrams, each activity will be given a unique number. The numbering convention will depend on how the various diagrams interrelate.</td>
</tr>
<tr>
<td><img src="image" alt="File" /></td>
<td>File</td>
<td>For DFD purposes a file is a temporary repository of data - a tape, a disk, an index file, etc. Data bases are illustrated as files.</td>
</tr>
<tr>
<td><img src="image" alt="Source or sink" /></td>
<td>Source or sink</td>
<td>A source or sink is a person or organization, outside of the system that is originator or receiver of system data. A person or organization inside the system would be characterized by the processes she/he or it performs.</td>
</tr>
</tbody>
</table>
The notation of Gane/Sarson differs a little bit in the use of the rectangle - a rectangle is used instead of a circle for illustrating processes because of the advantage that text can better be inserted.

Even if the control flow shall explicitly not be represented in a DFD, it could easily be illustrated e.g. by numbers or a special order of the elements. Those improvements would make DFDs better suitable for the modeling of business process models. (cp. Section 3.9)

‘Figure 30 is read as follows: X’s arrive from the source Q and are transformed into Y’s by the process P1 (which requires access to the file F to do its work). Y’s are subsequently transformed into Z’s by the process P2’ (DeMarco, 1979:51).
3.8.2. Demonstration of a data flow diagram (DFD)

Figure 31: Demonstration of a data flow diagram - Original notation

For understanding and preventing misunderstandings in a data flow diagram a data dictionary is obligatory for every diagram. Every data flow should be stated precisely like in the following example:

Payment Data = Customer-Name + Customer-Address + Invoice number + Amount of payment;

Invoice number = State-Code + Customer-Account-Number + Salesman-ID + Sequential-Invoice-Count; (cf. DeMarco, 1979:39)
3.8.3. Evaluation of the data flow diagram notation

Data flow diagrams are very easy to draw by hand because of the flexible data flow connectors. On the first view seem DFDs very confusing because of the missing order of the elements. Because of the fact that DFDs are just focusing on the data flow, loops, conditions and structured design are not applicable.

An idea for improvement would be advancement for the illustration of the control flow for increasing readability, clarity, clearness and logic. Please see Section 3.9.4 for more details.
DFDs have in their original notation just a minor importance and influence in business process modeling.

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>7</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>0</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>0</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>5</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>0</td>
</tr>
</tbody>
</table>

![Figure 32: Strengths and weaknesses profile of data flow diagrams](image)

Please see Section 3.3.4 for more information about the evaluation procedure.

3.8.4. Idea for improvement of data flow diagrams

The elements of a DFD could be easily supplemented with numbers, which represent the logical order of the elements and so the control flow. Another suggestion is the use of a certain order of the elements as illustrated in Figure 33.
I supplemented the demonstration example from Figure 31 with both of my improvement suggestions. With the help of numbers and letters is it also possible to represent parallelism of two actions, as you can see in Figure 34.
3.9. Swim lane diagram

Swim lane diagrams were originally called “Organisationsprozessdarstellung (OPD)” [Organizational process representation] and developed by H. F. Binner in the early 1980s. A swim lane is analogous to an area of responsibility for the process actors, where the dedicated responsibility is see-sawing in between until the procedure is completed. UML activity diagrams and the business process diagrams from Österle inherited swim lanes (cp. Gadatsch, 2005:62, translated by KG).

3.9.1. Notation of Swim lane diagrams

The notation of swim lane diagrams has been developed in different ways and can have different characteristics, depending if you are modeling a rough process model or a detailed workflow model.

In the simplest form, swim lanes are designed with just a few elements that make them very easy to draw. For workflow modeling, more precise elements are available e.g. control flow operators.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Process step symbol" /></td>
<td>Process step</td>
<td>Illustration of activities</td>
</tr>
<tr>
<td><img src="image" alt="Control flow symbol" /></td>
<td>Control flow</td>
<td>Temporal logical course of process steps supplemented with alternatives (yes, no) and probabilities in %</td>
</tr>
<tr>
<td><img src="image" alt="Alternative symbol" /></td>
<td>Alternative</td>
<td>Alternative in the course of activities</td>
</tr>
<tr>
<td><img src="image" alt="XOR symbol" /></td>
<td>Logical operator ‘exclusive OR’</td>
<td>For workflow models logical linking operators can be used additionally</td>
</tr>
<tr>
<td><img src="image" alt="OR symbol" /></td>
<td>Logical operator ‘OR’</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="AND symbol" /></td>
<td>Logical operator ‘AND’</td>
<td></td>
</tr>
</tbody>
</table>
3. Graphical notations for business process models

<table>
<thead>
<tr>
<th>Actor 1</th>
<th>Swim lane</th>
<th>Responsibilities of the actors (persons, departments, companies)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actor n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Document</th>
<th>Document, Database or Information object relating to the course</th>
</tr>
</thead>
</table>

Table 11: Notation swim lane diagrams (Gadatsch, 2005:63, translated by KG)

3.9.2. Demonstration of a swim lane diagram

![Swim lane diagram](image)

Figure 35: Demonstration of a swim lane diagram - Original notation

3.9.3. Evaluation of the swim lane diagram notation

Swim lane diagrams give a fast, clear and simple overview over the process responsibilities and the process course itself. The graphical notation is easy to draw by hand and supports all the important and required symbolism for BPM.
3. Graphical notations for business process models

### 3.9.4. Idea for improvement of swim lane diagrams

Structured symbolism for loops:
With the use of [[*]] the unstructured diagram (Figure 35) can be easily transformed into a structured diagram (Figure 37). The original notation includes no structured symbolism for loops.

![Figure 37: Demonstration of a swim lane diagram - Improved notation](image)

Please see Section 3.3.4 for more information about the evaluation procedure.

![Figure 36: Strengths and weaknesses profile of swim lane diagrams](image)
3.10. Block diagram

‘A block diagram is a specialized, high-level type of flowchart. Its highly structured form presents a quick overview of major process steps, as well as the relationships and interfaces involved.

A block diagram is a useful tool for designing new processes and in improving existing processes. In both cases, the block diagram provides a quick, high-level view on the process. Because of its high-level perspective, it may not offer the level of detail required for more comprehensive planning or analysis. Team members who construct a block diagram must have a clear understanding of how the process operates.’ (cp. Concordia [online])

3.10.1. Notation of block diagrams

Block diagrams consist of just two symbols:
Rectangles for representing the single process steps and
Arrows for connecting the process steps and representing the control flow
3.10.2. Demonstration of a block diagram

![Diagram showing the process of recording orders, checking orders, ordering products, creating order confirmations, sending order confirmations, creating invoices, shipping goods, sending invoices, checking payments, creating payment reminders, and closing orders.]

Figure 38: Demonstration of a block diagram - Original notation (left) and Improved notation (right)
3.10.3. Evaluation of the block diagram notation

Block diagrams represent the simplest graphical notation used for BPM. The diagram is very easy to understand and easy to draw by hand. Because of the fact that parallelism is not supported I had to arrange i.e. the activities “create invoice”, “ship goods” and “send invoice” as a sequence in the diagram.

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>10</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>0</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>0</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>0</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

Figure 39: Strengths and weaknesses profile of block diagrams

3.10.4. Idea for improvement of block diagrams

Structured symbolism for loops:

With the use of [ ] a the unstructured diagram (Figure 38, left) can be easily transformed into a structured diagram (Figure 38, right). The original notation includes no structured symbolism for loops.
3.11. Event-driven process chain

Event driven process chains (EPCs) were developed by Keller, Nüttgens, and Scheer and are representing a semi-formal notation for modeling business processes. Next to the description of business processes in the as-is state, EPCs are used for the design of process models in the target state. (cp. Heinrich 2003:403 and Staud, 2006:59, translated by KG)

EPCs are based on the concept of the directed graph (like Petri Nets, please see Section 3.6). Nodes are representing events; directed links are modeling the control flow.

3.11.1. Notation of EPCs

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Event" /></td>
<td>Event</td>
<td>Description of a occurred state from which the rest of the process course is dependent. Events are initiating functions or are the results of functions. Events are passive elements in the process chain.</td>
</tr>
<tr>
<td><img src="image" alt="Function" /></td>
<td>Function</td>
<td>Functions are active elements in the process chain. They can be defined as tasks or performances. Functions are changing the system state, meaning that they are processing or executing material and informational objects by reading, changing, deleting, or producing them. So the aim of the function is the transformation of input to output. Therefore, functions can make decisions and influence the following course of the process. Functions are linked by events.</td>
</tr>
<tr>
<td><img src="image" alt="Logical operator 'exclusive OR'" /></td>
<td>Logical operator ‘exclusive OR’</td>
<td>Logical linking operators are describing the logical linking of events and functions</td>
</tr>
<tr>
<td><img src="image" alt="Logical operator ‘OR'" /></td>
<td>Logical operator ‘OR’</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Logical operator ‘AND’" /></td>
<td>Logical operator ‘AND’</td>
<td></td>
</tr>
</tbody>
</table>

Table 12: Basic Elements of EPCs (Gadatsch, 2005:158, translated by KG)

Events and functions are connected by directed links which are represented by dashed lines.
3.11.2. Demonstration of an event driven process chain

Figure 40: Demonstration of an EPC diagram - Original notation
3.11.3. Evaluation of the EPC notation

EPC diagrams differ from the other types of graphical notations. Because of the fact that the EPC notation is based on functions and events I had to model several events like “order received”, “product on stock” or “14 days after invoice” for being able to illustrate my order example. Nevertheless are EPC diagrams easy to understand when the interpreting person knows about the function/event-system.

Small EPC diagrams are easy to draw by hand - for more complicated business processes I would recommend the use of special software like ARIS or MS Visio.

The EPC notation offers all the important symbolism required for BPM and so I had no problems in applying the notation to my example.

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbolismp for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>10</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>0</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

![Figure 41: Strengths and weaknesses profile of EPC diagrams](image)

3.11.4. Idea for improvement of EPC diagrams

Structured symbolism for loops:

With the use of [ ] the unstructured diagram (Figure 40) can be easily transformed into a structured diagram (Figure 42). The original notation includes no structured symbolism for loops.
Figure 42: Demonstration of an EPC diagram - Improved notation
3.12. PROMET diagram

PROMET (PROcess METHod) was developed by Österle in 1993. It connects the business strategy, the resulting processes, and the processes supporting information systems. The process modeling begins with a process chain diagram as a directed graph what is illustrated by different node and edge types. The diagram is structured by rows (swim lanes) for representing the organizational units or applications (cf. Gadatsch, 2005:84f, translated by KG).

3.12.1.1. PROMET-Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Event" /></td>
<td>Event</td>
<td>Trigger or result of a course</td>
</tr>
<tr>
<td><img src="image" alt="Process transition" /></td>
<td>Process transition</td>
<td>Transition from and to a process or sub-process (process interface)</td>
</tr>
<tr>
<td><img src="image" alt="Control flow (without delay)" /></td>
<td>Control flow (without delay)</td>
<td>Course without time delay meaning that the following task can start after the previous task is completed.</td>
</tr>
<tr>
<td><img src="image" alt="Control flow (with delay)" /></td>
<td>Control flow (with delay)</td>
<td>Task course with time delay meaning that the following task is allowed to start after the completion of the previous task with a time delay.</td>
</tr>
<tr>
<td><img src="image" alt="Control flow (parallel execution)" /></td>
<td>Control flow (parallel execution)</td>
<td>Tasks which are using this form of connector can be processed parallel to one another.</td>
</tr>
<tr>
<td>not applicable</td>
<td>not applicable</td>
<td>Tasks which are not connected with each other are concurrent, meaning they can be processed parallel or one after the other.</td>
</tr>
<tr>
<td><img src="image" alt="Document" /></td>
<td>Document</td>
<td>Document, relating to the course</td>
</tr>
<tr>
<td><img src="image" alt="Database access" /></td>
<td>Database access</td>
<td>Access of applications on database. DB-ID: Database Identification</td>
</tr>
</tbody>
</table>

Table 13: Notation of PROMET diagrams (Gadatsch 2005:84, translated by KG) supplemented with documents
3.12.2. Demonstration of a PROMET diagram

In PROMET diagrams, only the logical operator ‘exclusive OR’ is available and as you can see in Figure 43, the illustration by rows is not very adequate for complex models. Overlapping edges can result very easily.

Figure 43: Demonstration of a PROMET diagram - Original notation
3.12.3. Evaluation of the PROMET notation

PROMET diagrams include all required aspects for BPM, even parallelism. Next to the control flow also the data flow is included in the notation. PROMET diagrams are basically swim lane diagrams, supplemented with data flow and events. Even if the PROMET notation seems very unclear at the first view it gives us a possibility of combining data flow, control flow, loops, conditions and parallelism in one diagram. For drawing this diagram I used MS Visio, but I would highly recommend special software for this notation. The concept of different arrows for expressing control flow parallelism is a very good and simple approach.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarity and clearness</td>
<td>3</td>
</tr>
<tr>
<td>Easy to draw by hand?</td>
<td>3</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>5</td>
</tr>
<tr>
<td>Required space</td>
<td>3</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>10</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

**Figure 44: Strengths and weaknesses profile of PROMET diagrams**

3.12.4. Idea for improvement of PROMET diagrams

Structured symbolism for loops:

With the use of [] the unstructured diagram (Figure 43) can be easily transformed into a structured diagram (Figure 45). The original notation includes no structured symbolism for loops.
3. Graphical notations for business process models

Figure 45: Demonstration of a PROMET diagram – Improved notation
3. UML activity diagram

An UML activity diagram describes a course with the help of actions and edges. With this kind of diagram, use cases can be described in natural language and illustrated graphically. With natural language, normally only trivial courses can be documented clearly whereas with activity diagrams, it is possible to document very complex courses in a clear and comprehensible way.

The elements of an activity which illustrate actions are called activity nodes. An activity consists of activity nodes and activity edges. The activity edges are directed and connect the nodes. (cf. Staud, 2006:375 ff)

3.13.1. Notation of activity diagrams

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| ![Activity node](image) | Activity     | An action is a single step in a process course and is represented by a rectangle with rounded edges and with the action name in the middle. An action normally has ingoing and outgoing edges. Ingoing edges are starting an action. If there are more ingoing edges existent, the action is started when all ingoing edges are activated. At the finalization of an action all outgoing edges become active.
Activity nodes can include different behaviors e.g. a call for an operation or an execution. They can also include control flow elements for alternative or combination of edges when nodes are processed parallel. |
| ![Object node](image) | Object       | Object nodes are representing objects as we understand them from the object-oriented approach. The task of an object node is to model the flow of objects of an activity.                                                                                                                      |
| ![Control flow edge](image) | Control flow edge | The execution of a node leads to the execution of other nodes. The control is passed on from node to node on the edges: If an action is executed, the control is passed on to the next one, etc.                                                                                                   |
| ![Object flow edge (Data flow)](image) | Object flow edge (Data flow) | We can distinguish two different flows in activity diagrams: Control flows and object (data) flows. Both flows are representing different, but dependent aspects!                                                                                                              |
### Graphical notations for business process models

The distinction between control and data flows is strict: Every edge which is leading to or leading from object nodes has to be an object flow edge!

A signal represents a notification about an important incident which immediately activates an edge. Signals are illustrated by arrows. Particularly for time events, a sandglass symbol is used. Received signals are illustrated by an edge which points to the action which receives the signal. Sent signals are illustrated by an edge which is leading away from the action which sent the signal.

Swim lanes in an activity diagram are describing who or what is responsible for a certain activity. Swim lanes are illustrated by vertical lines. The elements of an activity diagram (activities, object nodes) are always exactly situated within one swim lane.

#### Table 14: Basic Elements of an UML activity diagram (cp. Staud, 2006:374ff, translated by KG)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Signal" /></td>
<td>Signal. A signal represents a notification about an important incident which immediately activates an edge. Signals are illustrated by arrows.</td>
</tr>
<tr>
<td><img src="image" alt="Swim lanes" /></td>
<td>Swim lanes. Swim lanes in an activity diagram are describing who or what is responsible for a certain activity. Swim lanes are illustrated by vertical lines. The elements of an activity diagram (activities, object nodes) are always exactly situated within one swim lane.</td>
</tr>
<tr>
<td><img src="image" alt="Decision Node/Merge node" /></td>
<td>Decision Node/Merge node (Alternative/Exclusive OR). Control nodes</td>
</tr>
<tr>
<td><img src="image" alt="Fork Node/Join Node" /></td>
<td>Fork Node/Join Node (AND).</td>
</tr>
<tr>
<td><img src="image" alt="End node - Flow Final" /></td>
<td>End node - Flow Final. An end node marks the end of a course in an activity diagram. They are illustrated by a filled circle with a ring around it.</td>
</tr>
<tr>
<td><img src="image" alt="End node - Activity Final" /></td>
<td>End node - Activity Final. An activity end node is illustrated by a circle with a cross in it. Activity end nodes and final end nodes have at least one ingoing, but no outgoing edges.</td>
</tr>
<tr>
<td><img src="image" alt="Start node - Initial node" /></td>
<td>Start node - Initial node. A start node marks the start of a course and it has outgoing, but no ingoing edges.</td>
</tr>
</tbody>
</table>

An activity diagram consists of a start- and an end-node as well as some actions and edges which are connecting those elements together. In- and outgoing objects are called parameters of the activity. Edges are describing the transition of an action to the next one and so the control- and
object flow as continuous lines. Simple arrows point on the following action. On the edges, we can find conditions in angled bracelets.

Normally, an activity diagram has only one start-, but several end nodes. If there are more start nodes existent, they are normally starting their own concurrent edges. If there is more then one end node, each of these end nodes brings the concurrent edges immediately to an end over the entire diagram (cp. Oesterreich et al, 2004:175ff).

### 3.13.2. Demonstration of an activity diagram

![Figure 46: Demonstration of an UML activity diagram - Original notation](image)

Karin Margarete Grünauer
**3.13.3. Evaluation of the UML activity diagram notation**

UML activity diagrams are easy to understand, easy to draw by hand and offer all the required symbolism needed for good business process modeling. With an UML activity diagram even complicated processes can be illustrated in a clear way. UML activity diagrams got the best results in my evaluation in comparison to the other evaluated graphical notations.

Only one small idea for improvement I have to point out: a distinction between the control flow and data flow connectors. Please see Section 3.14.4 for more details.

<table>
<thead>
<tr>
<th>Clarity and clearness</th>
<th>10</th>
<th>Clarity and clearness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>10</td>
<td>Easy to draw by hand?</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>10</td>
<td>Easy to understand?</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
<td>Required space</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
<td>Structured Design mandatory?</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
<td>Structured symbolism for loops included?</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
<td>Symbolism for conditions included?</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>10</td>
<td>Parallelism supported?</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
<td>Data flow aspect supported?</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
<td>Control flow aspect supported?</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

*Figure 47: Strengths and weaknesses profile of UML activity diagrams*

**3.13.4. Idea for improvement of UML activity diagrams**

1.) Distinction between control flow and data flow arrows:
As you can see in Figure 48, I improved the UML activity diagram from Figure 46 in using dashed arrows for a representation of the data flow. In the original notation is no distinction between control flow and data flow arrows made.

2.) Structured symbolism for loops:
With the use of [] the unstructured diagram (Figure 46) can be easily transformed into a structured diagram (Figure 48). The original notation includes no structured symbolism for loops.
Figure 48: Demonstration of an UML activity diagram – Improved notation
3.14. Extended event driven process chains (eEPC)

The original EPC notation is recommended for a simple illustration of the control flow. For a more advanced illustration, the connection of all views on the business process is missing. In 1998 the original EPC notation got some advancement and is called extended EPC. Functions require not only events which are triggering courses, but they also need inputs (especially information, but material as well). For the analysis and the design of business processes, not only the static structure of data is important, also the dynamic view, especially when we are talking about information. Also essential for the optimal information supply is information about where and from which position or function in the company the information objects are accessed or changed.

(cp. Gadatsch, 2005:82f, translated by KG)

This is the reason why the extended event driven process chain includes information objects in the notation. Information objects show which information in the process course is used by which function. The information objects are illustrated by rectangles. A drawn through arrow, directed on a function means the usage (e.g. reading) of information by a function. An arrow, directed on an information object means the processing or the changing of the information object by a function.

(Heinrich, 2003:405, translated by KG)

3.14.1. Notation of eEPC diagrams

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Event" /></td>
<td>Event</td>
<td>Description of a occurred state from which the rest of the process course is dependent</td>
</tr>
<tr>
<td><img src="image" alt="Function" /></td>
<td>Function</td>
<td>Description of the transformation of an input state to an output state</td>
</tr>
<tr>
<td>![Logical operator 'exclusive OR'](</td>
<td>Logical operator 'exclusive OR'</td>
<td>Logical linking operators describe the logical linking of events and functions</td>
</tr>
<tr>
<td>![Logical operator 'OR'](</td>
<td>Logical operator ‘OR’</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Logical operator ‘AND’" /></td>
<td>Logical operator ‘AND’</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td><img src="image1" alt="Organizational unit" />.</td>
<td>Organizational unit</td>
<td>Description of the structure of a company</td>
</tr>
<tr>
<td><img src="image2" alt="Information object" />.</td>
<td>Information object</td>
<td>Representation of information objects</td>
</tr>
<tr>
<td><img src="image3" alt="Control flow" />.</td>
<td>Control flow</td>
<td>Course of events and functions</td>
</tr>
<tr>
<td><img src="image4" alt="Data flow" />.</td>
<td>Data flow</td>
<td>Description of a function if it is read, written, or changed</td>
</tr>
<tr>
<td><img src="image5" alt="Assignment" />.</td>
<td>Assignment</td>
<td>Assignment of resources/organizational units</td>
</tr>
<tr>
<td><img src="image6" alt="Process guidepost" />.</td>
<td>Process guidepost</td>
<td>Process refinement</td>
</tr>
</tbody>
</table>

Table 15: Basic Elements of eEPCs (Gadatsch, 2005:83, translated by KG)
3.14.2. Demonstration of an extended EPC

Figure 49: Demonstration of an extended event driven process chain (eEPC) - Original notation
3.14.3. Evaluation of the eEPC notation

The extended EPC notation offers all the required symbolism for good BPM. With a little bit of knowledge about the notation, eEPC diagrams are easy to read and to understand. Original EPC diagrams are very easy to transform into the extended notation. For modeling processes with the eEPC notation, I recommend the use of special software like ARIS.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarity and clearness</td>
<td>8</td>
</tr>
<tr>
<td>Easy to draw by hand?</td>
<td>7</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>10</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

![Strengths and weaknesses profile of eEPC diagrams](image)

Please see Section 3.3.4 for more information about the evaluation procedure.

Figure 50: Strengths and weaknesses profile of eEPC diagrams

3.14.4. Idea for improvement of eEPC diagrams

Structured symbolism for loops:
With the use of []* the unstructured diagram (Figure 49) can be easily transformed into a structured diagram (Figure 51). The original notation includes no structured symbolism for loops.
Figure 51: Demonstration of an extended event driven process chain (eEPC) - Improved notation
3.15. Business process diagram (BPD)

(cp. Gadatsch, 2005:85, translated by KG)

Business process diagrams are basically directed and labeled graphs with different types of nodes:
Business process
Activity authority
Documents
Data memory and
Logical linking operators.

Next to those types of nodes, a business process diagram consists of different types of edges:

Control flow
Information flow
Data flow and
Organizational allocation.

One type of edge always connects two types of nodes.

The control flow connects the node type “business process” and is split up by the logical linking operator. Furthermore, the control flow represents events.

An organizational allocation connects the business process with the activity authorities.

The edge type information flow connects the node type business process with the data memory.

A business process diagram starts with at least one event and ends with at least one event. As you can see in Figure 52, business process diagrams offer three different process views:

1. Organizational view for representing the involved activity authorities.
2. Informational view for the illustration of the required information objects.
3. Activity view for the illustration of the business process and its required applications from a process oriented point of view.

A big advantage of business process diagrams is that they can be easily refined into workflow models.
### 3.15.1. Notation of BP diagrams

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Organization" /> <img src="image" alt="Organizational unit" /></td>
<td>Activity authority</td>
<td>Organization (e.g. company, sub company) or organizational unit (e.g. department, project)</td>
</tr>
<tr>
<td><img src="image" alt="BP-ID" /> <img src="image" alt="BP name" /> <img src="image" alt="IS" /> <img src="image" alt="IS Application" /> <img src="image" alt="BP/WF" /></td>
<td>Business process or business process step</td>
<td>BP-ID: Identification of the business process in the repository IS: Name of the information system BP: Business process or Business process step which, if necessary, is further refined WF: Business process or Business process step which is refined as Workflow</td>
</tr>
<tr>
<td>— — — ➔</td>
<td>Organizational assignment</td>
<td>Activity authority controls the dedicated business process and is also the process owner.</td>
</tr>
<tr>
<td>➔</td>
<td>Control flow</td>
<td>Linking of objects “events”, “business process” and the linking operator.</td>
</tr>
<tr>
<td>XOR ∧ V</td>
<td>Linking operators</td>
<td>Exclusive OR, AND, OR</td>
</tr>
<tr>
<td><img src="image" alt="Data memory name" /></td>
<td>Data memory</td>
<td>Automatically processed information carrier. Data memory name: Identifying name in the repository</td>
</tr>
<tr>
<td><img src="image" alt="Document" /></td>
<td>Document</td>
<td>Document, relating to the course</td>
</tr>
</tbody>
</table>

Table 16: Notation of BP diagrams (Gadatsch, 2005:86, translated by KG) supplemented with documents
### 3.15.2. Demonstration of a BP diagram

#### Business process diagram (BPD)

<table>
<thead>
<tr>
<th>Organizational view</th>
<th>Activity view</th>
<th>Information view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shipping plc. Sales department</td>
<td><strong>BPI-CHOR</strong> Check order - products on stock</td>
<td><strong>BPI-CSRE</strong> Create and send invoice</td>
</tr>
<tr>
<td>Shipping plc. Distribution</td>
<td><strong>BPI-CROC</strong> Create and send order confirmation</td>
<td><strong>PC</strong> Word processing</td>
</tr>
<tr>
<td>Shipping plc. Accounts department</td>
<td><strong>SAP R/3</strong></td>
<td><strong>BPI-ORRP</strong> Order product</td>
</tr>
<tr>
<td>Shipping plc. Sales department</td>
<td><strong>Module MM</strong></td>
<td><strong>SAP R/3</strong></td>
</tr>
<tr>
<td></td>
<td><strong>BPD</strong></td>
<td><strong>Module SD</strong></td>
</tr>
<tr>
<td></td>
<td><strong>BPD</strong></td>
<td><strong>BPI-CHPA</strong> Check payment</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Module FI</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BPI-CLOR</strong> Close offer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SAP R/3</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Module SD</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BPD</strong></td>
</tr>
</tbody>
</table>

**Figure 52: Demonstration of a BPM diagram - Original notation**

### 3.15.3. Evaluation of the BPD notation

The BPD notation tries to illustrate every little aspect in one diagram. As you can see in Figure 52, the diagram is not very easy to read on the first impression. In applying my order example to the notation I got the impression that already with my simple business process the notation is reaching its limits. For a very detailed and refined illustration of business processes the notation can be thoroughly practical because of the high amount of information in just one diagram. For this notation I highly recommend the use of special software.

### 3.15.4. Idea for improvement of BP diagrams

Structured symbolism for loops:

With the use of $[]^*$ the unstructured diagram (Figure 52) can be easily transformed into a structured diagram (Figure 54). The original notation includes no structured symbolism for loops.
3. Graphical notations for business process models

| Clarity and clearness | 2 |
| Easy to draw by hand? | 2 |
| Easy to understand? | 2 |
| Required space | 2 |
| Structured Design mandatory? | 5 |
| Structured symbolism for loops included? | 0 |
| Symbolism for conditions included? | 10 |
| Parallelism supported? | 10 |
| Data flow aspect supported? | 10 |
| Control flow aspect supported? | 10 |

Please see Section 3.3.4 for more information about the evaluation procedure.

**Figure 53: Strengths and weaknesses profile of BP diagrams**

**Figure 54: Demonstration of a BP diagram - Improved notation**
3.16. Object oriented event driven process chain (oEPC)

In 1997, Scheer developed an object oriented EPC notation for reaching two goals:

Application of the event driven concept of object interaction in object-oriented, component-based information systems, and

Integrated description of processes and their objects.

The object-oriented paradigm requires a modification of the term “business process”, which describes the event driven processing of business objects for goods and services. Business objects encapsulate functions (methods) and data (instance variables), which are required for creating benefits.

Events are not starting functions, like we learned in the definition of the classical EPC (cp. Section 3.12). In the object-oriented approach of EPCs, events start a call for methods of the instantiated objects. The notation is only based on the classical EPC notation! New symbols for business objects (business class), data (instance variables) and functions (methods) are introduced.

(cp. Gadatsch, 2005:97, translated by KG)

### 3.16.1. Notation of oEPCs

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Naming</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Object class" /></td>
<td>Object class</td>
<td>Business object which encapsulates relevant functions (methods) and data (instance variables) for processing</td>
</tr>
<tr>
<td><img src="image" alt="Event/message" /></td>
<td>Event/message</td>
<td>Description of a occurred state from which the rest of the process course is dependent</td>
</tr>
<tr>
<td><img src="image" alt="Method/function" /></td>
<td>Method/function</td>
<td>Function (method) of an object for manipulating data (instance variable). Private methods are not visible outside of the object.</td>
</tr>
<tr>
<td><img src="image" alt="Instance var. /Attribute" /></td>
<td>Instance var. /Attribute</td>
<td>Data (Instance variable) which are manipulated by the methods of an object</td>
</tr>
<tr>
<td><img src="image" alt="Organizational unit" /></td>
<td>Organizational unit</td>
<td>Description of the structure of a company</td>
</tr>
</tbody>
</table>
### 3. Graphical notations for business process models

<table>
<thead>
<tr>
<th>Connector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>Logical linking operator for the linking of business objects and events.</td>
</tr>
<tr>
<td>△</td>
<td>Control flow</td>
</tr>
<tr>
<td>←→</td>
<td>Temporal-logical dependency of events and business objects</td>
</tr>
<tr>
<td>←→</td>
<td>Service relation</td>
</tr>
<tr>
<td></td>
<td>Event driven messages between business objects</td>
</tr>
<tr>
<td>————</td>
<td>Edge</td>
</tr>
<tr>
<td></td>
<td>Assignment of methods, instance variables and organizational units to objects</td>
</tr>
</tbody>
</table>

Table 17: Notation of oEPC (Gadatsch, 2005:98, translated by KG)
3.16.2. Demonstration of an oEPC

Figure 55: Demonstration of an object oriented EPC - Original notation
The object-oriented notation of EPCs enables an integrated description of objects and processes which is not possible in the classical EPC notation. However, the object-oriented EPC could still never reach the level of success of the classical notation in practice.

### 3.16.3. Evaluation of the oEPC notation

The main focus of the oEPC notation doesn’t lie on functions (object methods) and so the notation requires an identification of business objects. Therefore might the notation be harder to understand for people who have no knowledge about object orientation.

For applying my standardized order example to the oEPC notation, I used the orginal EPC diagram (Figure 40 and 42) and transformed it into the oEPC notation. Normally, existent ‘classical’ EPC diagrams can be easily transformed into an object oriented EPC. The basic structure barely changes. The object class replaces the event and the function, which is assigned as method to the object class.

<table>
<thead>
<tr>
<th>Clarity and cleanness</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to draw by hand?</td>
<td>5</td>
</tr>
<tr>
<td>Easy to understand?</td>
<td>8</td>
</tr>
<tr>
<td>Required space</td>
<td>5</td>
</tr>
<tr>
<td>Structured Design mandatory?</td>
<td>5</td>
</tr>
<tr>
<td>Structured symbolism for loops included?</td>
<td>0</td>
</tr>
<tr>
<td>Symbolism for conditions included?</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism supported?</td>
<td>10</td>
</tr>
<tr>
<td>Data flow aspect supported?</td>
<td>10</td>
</tr>
<tr>
<td>Control flow aspect supported?</td>
<td>10</td>
</tr>
</tbody>
</table>

Please see Section 3.3.4 for more information about the evaluation procedure.

**Figure 56: Strengths and weaknesses profile of oEPC diagrams**

### 3.16.4. Idea for improvement of oEPC diagrams

Structured symbolism for loops:

With the use of \( []^* \) the unstructured diagram (Figure 55) can be easily transformed into a structured diagram (Figure 57). The original notation includes no structured symbolism for loops.
3. Graphical notations for business process models

Figure 57: Demonstration of an object oriented EPC - Improved notation
3.17. Analogy of the elementary components of control flow models and business process models

After the overview over the different notations, I want to point out the analogies of control flow models and business process models. ‘Control flow models and business process models have analogous elementary components although they model different situations and use different notations. Both are behavioral models as they represent courses of activities and events. While the main aspect of the programmer is the source code which he has to write, the main focus of the business process model designer are the enterprise and the business process which he has to represent. Although both of them use different perspectives, we can figure out analogous elementary components in either modeling approach’ (Holl, 2005).

The following examples are taken from Alfred Holl’s and Gregor Valentin’s paper “Structured business process modeling (SBPM)” (2005):

- The instructions of the programmer are the business activities of the business process model designer. We can call this type of elementary components process units using an umbrella term on a more abstract level.
- Events in a company are caused from outside of the company, e.g. a phone call from a customer, or from inside a company when an activity is finished. For programmers, an event is always induced by the operating system.
- For a business process modeler, iteration means to repeat the same business activities in the same order for several times in a cycle. For the programmer, this means a repetition of the same program instructions in a loop.

Alfred Holl and Gregor Valentin (2005) describe furthermore that it is possible to compare all of the other elementary components of either modeling approach:

<table>
<thead>
<tr>
<th>Umbrella term</th>
<th>BPM</th>
<th>Control flow modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular sub-structure</td>
<td>Partial process</td>
<td>Sub-program, sub-routine</td>
</tr>
<tr>
<td>Sequence</td>
<td>Sequence</td>
<td>Sequence</td>
</tr>
<tr>
<td>Test, alternative, decision</td>
<td>XOR</td>
<td>IF</td>
</tr>
<tr>
<td>Iteration</td>
<td>Cycle</td>
<td>Loop</td>
</tr>
<tr>
<td>Event</td>
<td>Business event</td>
<td>Operating system event, interrupt</td>
</tr>
<tr>
<td>Process unit</td>
<td>Business activity</td>
<td>Instruction or block of instructions</td>
</tr>
<tr>
<td>Simultaneity</td>
<td>AND</td>
<td>Parallel functions</td>
</tr>
</tbody>
</table>

Table 18: Analogy of the elementary components of BPM and control flow modeling (Holl, 2005)
<table>
<thead>
<tr>
<th>Umbrella term</th>
<th>Structure diagram</th>
<th>Control flow chart</th>
<th>eEPC</th>
<th>UML activity diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular sub-structure</td>
<td><img src="image" alt="Modular sub-structure" /></td>
<td><img src="image" alt="Control flow chart" /></td>
<td><img src="image" alt="eEPC" /></td>
<td>No symbol</td>
</tr>
<tr>
<td>Event</td>
<td>No symbol</td>
<td>No symbol</td>
<td><img src="image" alt="Event" /></td>
<td>No symbol</td>
</tr>
<tr>
<td>Sequence</td>
<td><img src="image" alt="Sequence" /></td>
<td><img src="image" alt="Control flow chart" /></td>
<td><img src="image" alt="eEPC" /></td>
<td><img src="image" alt="UML activity diagram" /></td>
</tr>
<tr>
<td>Alternative/Decision</td>
<td><img src="image" alt="Alternative/Decision" /></td>
<td><img src="image" alt="Control flow chart" /></td>
<td><img src="image" alt="eEPC" /></td>
<td><img src="image" alt="UML activity diagram" /></td>
</tr>
</tbody>
</table>
### Table 19: Analogy of the graphical notations of BPM and control flow modeling

<table>
<thead>
<tr>
<th>Iteration:</th>
<th>Old: unstructured</th>
<th>New: structured</th>
<th>Parallelism</th>
<th>Process unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO-WHILE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPEAT- UNTIL, WHILE,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOR (x = )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a to b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WHILE C do</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPEAT UNTIL C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- **Old: unstructured**
- **New: structured**
- **Parallelism**
- **Process unit**

Karin Margarete Grünauer
Alfred Holl and Gregor Valentin point out that ‘both of the modeling approaches differ not only in the names of their elementary components. They also use different notations’ (Holl, 2005).

‘Table 19 shows that each elementary component of control flow modeling has its analogous counterpart in BPM. In addition, business process models can contain non-temporal information: the essential components describing the mere course over time can be extended by roles (actors), such as external partners and responsible departments/persons, and by data stores and resources used. But these components are only accidental as they do not affect the temporal structure of a process/behavioral model. As a result, we can state that all of the features of structured programming (cf. Section 2) can be transformed to BPM.

Furthermore, Table 19 contains a striking argument for the equivalence of control flow models and business process models. The umbrella terms of elementary process components in Table 19 can be regarded as a core meta-model of process/behavioral models in general (a meta-model is a list of all possible elementary components which can be used to establish models of a specific type). It describes possible components of process models in terms of a formalized natural language and is therefore independent of any notation, such as UML’ (Holl, 2005).
4. Business process decomposition under the aspects of Theory of Gestalt

This section of my master thesis discusses the motivated decomposition of business process models on several levels of abstraction. Theory of Gestalt plays an important roll for this approach. After an description of Theory of Gestalt and an precise definition of the term feature I will introduce Alfred Holl’s feature oriented EPC.

Overview:

4.1. Business process decomposition.................................105
4.2. Theory of Gestalt .........................................................106
4.3. Theory of Gestalt, processes, and their decomposition ....107
4.5. Target state of business process modelling...............109
4.6. Business process representation .................................111
Holl points out in his paper “Geschäftsprozessdekomposition und Gestalttheorie [Business process modeling and Theory of Gestalt]” that an essential part of business process modeling is decomposition on more than one level of abstraction. But those business process models are often not transparent and so not discussable. That’s the reason why software engineers reason from experience about different possibilities of refinement of a business process. The reason is that decomposition is an unconscious, creative act of every single model designer.

An improvement of this situation can be reached by application of the results of Theory of Gestalt. Decomposition is done with the help of features. Decomposition criteria of business process modeling also have explicit features in terms of Theory of Gestalt which lead to a better conscious traceability of business process models.

### 4.1. Business process decomposition

Business processes can be decomposed on several levels of cognition/perception.

![Figure 58: Structure of a process (Holl, 2000)](image)

When we are talking about decomposition of business processes, we can talk about taxonomical decomposition (specialization) on one side and/or compositional (temporal segmentation) on the other. Figure 58 shows the taxonomical decomposition of a production process in three versions for three different product groups. The decomposition is normally done simultaneously for both ways of decomposition.
4.2. Theory of Gestalt

4.2.1.1. Definition

‘Theory of Gestalt deals with human cognitive faculty and the features of human cognition. Gestalts are (complex) open systems (things, figures, objects, business processes etc.) which are connected and interpreted by the human cognitive apparatus. The main conclusion of Theory of Gestalt is that the entirety of a gestalt is more than its separate parts. We call this the “attribute of entirety”.

So-called phenomena of gestalt are mostly illustrated by optical illusions. But phenomena of gestalt are not only limited to optical cognitive processes, but include all sensory perceptions. The observer automatically interprets a subjective model (head of a horse) during the cognitive process (black and white spots). This leads us to some questions: How does perception work and are there any laws to follow? Theory of Gestalt deals with questions like this and tries to give us answers’ (Holl, 2000, translated by KG).

Figure 59: Gestalt of a head of a horse (Holl, 2000)

‘The Theory of Gestalt is an interdisciplinary general theory which is the basis for different psychological findings and their appliance. The human is seen as an open system; he is interacting actively with his environment. Theory of Gestalt is a special approach for understanding the construction of structure in mental processes. It finds its origins in approaches from Wolfgang von Goethe, Ernst Mach and especially Christian von Ehrenfels and in the research activities of Max Wertheimer, Wolfgang Köhler, Kurt Koffka and Kurt Lewin’ (Holl, 2000, translated by KG).
Holl further points out that as result of the gestalt theoretical research, laws of gestalt were formulated. Those rules describe phenomena of gestalt perception. One of the most important gestalt rules is called the “tendency to simple forms” [Tendenz zur Guten Gestalt], which is also called “tendency of clearness” [Prägananztendenz].

‘Tendency to simple forms’ refers to the human ability of structuring objects. In our example, the observer thinks that the white surfaces are not in their position by accident - the observer thinks that they are following a certain law and that they have a special meaning.

‘This process of structuring or interpretation happens partially unconsciously. The ability of gestalt perception is subjective and differs from human to human. It cannot be taught and the previous knowledge of the observer plays an important roll. Cultural influences are also very important’ (cf. Holl, 2000, translated by KG).

### 4.3. Theory of Gestalt, processes, and their decomposition

‘If we have to split up a gestalt to minimize its complexity, our human perception helps us in finding the right spot for this decomposition. This is also applicable to business processes. If you see a business process as a gestalt you can easily see its dynamic behavior. Normal examples for Theory of Gestalt are static and can be easily processed by the “optic-tangible” human brain. The human brain has great difficulties with the decomposition of flows and their dynamic behavior. Just think of the decomposition of a melody, a movie, a theatre piece, a movement, or a course of pictures (as a graphical representation of a process)’ (Holl, 2000, translated by KG).

![Figure 60: Man’s face to a female body (Holl, 2000)](image)

The face of the man is changing to a female body from left to right. In which picture is the transformation from the man to the woman happening? Where is the best place to draw a boundary?

To understand the introduced approach of business process modeling, we have to think about how models are created. The combination of observation (empiristic part) and previous knowledge in form of reference models (rationalistic part) makes us create a model of the regarded area in our
imagination. This idea of imagination is influenced by phenomena, which are described by the laws of gestalt. The designer of a model decomposes it, creates groupings and a hierarchy of the model mostly unconsciously during observation. The created imagination of a model is going to be formalized and visualized afterwards. This happens with the help of formal languages, e.g. UML (Unified Modeling Language) or EPC (event driven process chains).

### 4.4. Business process modeling and decomposition

Business process modeling copes with two basic decisions (cp. Holl, 2000):

Decision about used concepts of classification (e.g. The search for decomposition criteria, in the following context called features) and

Decision about the level of aggregation.

The following considerations refer to decomposition (temporal segmentation). The described approach is also applicable to taxonomical segmentation.

‘The result of the decomposition of a business process is a strictly hierarchic structure. A process is split into n sub-processes (tree structure). There is no standardized approach for creating this structure. The structuring of business process models is affected by the imagination and the previous knowledge of each of the model constructers. Different model designers have a different view of the real subject and so they decompose the business processes using different criteria. Those (decomposition-) criteria are not, or just partially, externalized. Without knowing those unconscious, implicit, and mostly undocumented thoughts, ideas and constraints, it is hardly possible to understand a business process model in detail. This state is unsatisfying’ (Holl, 2000, translated by KG).

#### 4.4.1. Comparison with normalized data modeling

In phase 1 of modeling, a model of the current situation is created in the mind of the model designer, based on observation and previous knowledge (data model: attributes, candidate keys, entity-types and relationships; business process model: processes, sub-processes and events).

In the following phase of modeling, there are big differences between data modeling and business process modeling. With the help of normalization in data modeling, the created model can be transformed to a comprehensible, widely standardized form, but in business process modeling we have to be satisfied with the found decomposition from the previous step.
4.5. Target state of business process modeling

To a large extent, our goal is to make business process models comprehensible and standardizable, to remove their disadvantages or at least to minimize them. But how can we reach this goal? To do this, we will introduce the term “feature” from Theory of Gestalt (4.5.1) and link it to business process decomposition (4.4). The relation of features and events will be explained and will lead us to an extended illustration of business processes (4.5). Features can be optimized and will also be found in reference models (4.6).

4.5.1. Features

‘If we perceive something special, for instance an attribute of a figure or an activity, this specialty or attribute is called a feature.’ You can use features for grouping and classifying figures and activities. Rupert Riedl systemized the term of “feature”. He based his work on categories of attributes used by biologists for classifying creatures. Riedl showed that it is not only possible to use features to decompose static figures but also to subdivide into flows. He asked some people to find the best suitable point for decomposition in a continuously changing row of features’ (Holl, 2000, translated by KG). You can see one of those rows of features and the result of the experiment in Figure 61.

![Figure 61: Establishment of feature based boundaries (Riedl 1987:158f, translated by KG)](image-url)
Each feature of the row consists of several features, in this case four single features. An assembled feature is called a “complex feature”. The hatched bars show the boundaries found ad hoc. Another important circumstance Riedl mentioned is the “changing of the feature category”.

‘The term feature is defined in more detail in biology. But the introduced, reduced approach is satisfying for our purpose. What is not satisfying is that there is no determination between the feature and its feature value. For example: it is possible to say that the feature of a car is its green color. But this way of speaking is not suitable for our purposes. For our studies, it is more important to talk about features and feature values. In this example we would say: The car has the feature color with the feature value green. We will continue using this manner of speaking in the following paragraphs’ (Holl, 2000, translated by KG).

### 4.5.2. Relations between features and events

(cf. Holl, 2000, translated by KG)

Features and events are in direct relation with each other. An event happens if, and only if, a feature gets a new value. With this agreement, events won’t be chosen on different levels anymore, they correspond within a process with exactly one feature - the decomposition feature. This circumstance is the equivalent to a mathematical step function (cp. Figure 62).

![Figure 62: Sub-processes shown as a mathematical step function (Holl, 2000, translated by KG)](image_url)
4.6. Business process representation

Business processes are visualized in different ways in literature. Holl takes Scheer’s notation and enlarges it with features as decomposition criteria. The feature and its particular value are inserted in the small box next to the corresponding sub-process. Events are changes of the feature value (Figure 63).

‘A notation like this is desirable, because, with the help of the feature, it is obvious why the decomposition took place at exactly this point/position. So it is possible to discuss if the feature and the decomposition based on it is reasonable or not. We call this notation feature-oriented event driven process chain’ (Holl, 2000, translated by KG).

Figure 63: Diagram of a process with sub-processes, events and features (Holl, 2000, translated by KG)

4.6.1. Evolution of feature

‘The question of which are the (optimal) features for decomposing processes depends on the current process. This question can not be answered in general.

In discussing the quality of features, it is possible to improve the decompositions by iteration. The knowledge which feature is better for special decompositions creates profit for further model building processes’ (Holl, 2000, translated by KG).
5. Conclusion
Nowadays BPM is a big topic for every organization to improve and optimize their processes for creating higher effectiveness and larger profits. Graphical notations which were originally created for software design are used for BPM. This development led to a jumble of graphical notations and a sometimes inadequate adaptation of those notations to BPM.

A classification and evaluation of the different graphical notations within BPM was needed. The basis for this classification is a genealogical tree (cp. Section 3.1) of the most used graphical notations within BPM. The tree visualizes the development and the relations of the graphical notations. I came to the result that every graphical notation developed after 1962 represents a successor of three “father notations”: Control flow charts, the network modeling technique (NMT) and Petri nets. The notations and their symbolism is described in Section 3. After the general description I furthermore illustrated the use of every mentioned notation with the help of a standardized example. The advantages and disadvantages became obvious after this process. This is the reason why I decided to visualize them in a strengths- and weaknesses profile for every mentioned notation.

Those strengths- and weaknesses profiles enable the comparison of the different graphical notations especially regarding structured BPM, symbolism, simpleness and readability. All the described notations were illustrated in a structured and unstructured way in order to convince about the advantages of a structured BPM.

Structured software design became popular in the 1970s and its advantages are clear. But why is BPM largely still happening unstructured? This is due to the different views in Computer science and Information systems. The contact between those two branches is not close enough to generally recognize the similarity between BPM and software design. None of the current graphical notations automatically leads to structured models!

In order to visualize the defects caused by unstructured BPM I transformed an unstructured BP model into a structured model in my example in Section 2.2.

The advantages of SBPM can be summarized as follows:

- Business process models become more transparent
- The modification and adaptation of BP models becomes easier
- Business process reengineering (optimization) is done in a more effective way
- The mapping of BP models to WFMS becomes more effective

For a successful SBPM block structures should be used instead of mere control flow lines as well as hierarchically nested structures (LIFO principle) should be used instead fo overlapping structures.
Last but not least a motivated hierarchic modular structure (in accordance to subroutines) should be used to decompose processes vertically.

BPM notations should be urgently extended by symbols for iterations. Within structured programming, iterations have turned out to be a very important elementary component for control flow models. In order to convince about this fact, I illustrated every example with an unstructured and a structured symbolism for loops. Detailed information to the requirements to SBPM is found in Section 2.3.

The third big part of my master thesis discussed the motivated decomposition of BP models on several levels of abstraction.
In order to minimize the complexity of BP models they have to be decomposed. This process is highly affected by the imagination and the previous knowledge of the model designer. Those unconscious ideas, constraints and thoughts are mostly undocumented, which makes the comprehension of a decomposed BP model complex. I discussed this problem in Section 4 and described the aspects of Theory of Gestalt, which led us to the definition of the term “feature”. Those features help us to trace back the decision why decomposition happened at a specific position. I presented a great approach for this matter in Section 4.6: Alfred Holls “feature oriented EPC” which supplements the original EPC notation with features.

Växjö, June 2008

Karin Margarete Grünauer
6. Bibliography & Indexes
6.1. Books


### 6.2. Papers


Holl, Alfred; (2008). Epistemology and methodology in enterprise modeling and information systems modeling, Georg Simon Ohm University of Applied Sciences, Nuremberg, Germany, 11.02.08/1
6.3. Online references


California State University - Department of Computer Science, Control node examples of activity diagrams, [Online], Available: http://csci.csusb.edu/dick/cs372/07activity3.gif [2007-10-19]


6.4. Index of Tables

Table 1: Model versus reality (Holl, 2000, translated by KG) ..............................................17
Table 2: Different model types and notations ........................................................................32
Table 3: Control Flow Chart (DIN66001, 1966) ................................................................33
Table 4: Table of activities ......................................................................................................43
Table 5: Elements of Structure Diagrams (Gadatsch, 2005:80, translated by KG) ..............48
Table 6: Elements of Structure Diagrams (Gadatsch, 2005:80, translated by KG) ..............49
Table 7: Elements of Structure Diagrams (Kubicek and Kamin, 2005, [online] translated by KG) 54
Table 8: Notation IDEFO diagram (Gadatsch, 2005:70, translated by KG) ..........................58
Table 9: Notation IDEFO3 diagram (Gadatsch, 2005:70f, translated by KG) .........................60
Table 10: Notation Data Flow Diagram (DeMarco, 1979:51ff) ................................................63
Table 11: Notation swim lane diagrams (Gadatsch, 2005:63, translated by KG) ..................66
Table 12: Basic Elements of EPCs (Gadatsch, 2005:158, translated by KG) ......................66
Table 13: Notation of PROMET diagrams (Gadatsch 2005:84) suppl. with documents .........66
Table 14: Basic Elements of an UML activity diagram (cp. Staud, 2006:374ff, translated by KG) 66
Table 15: Basic Elements of eEPCs (Gadatsch, 2005:83, translated by KG) .......................66
Table 16: Notation of BP diagrams (Gadatsch, 2005:86) supplemented with documents ......66
Table 17: Notation of eEPC (Gadatsch, 2005:98, translated by KG) .....................................66
Table 18: Analogy of the elementary components of BPM and control flow modeling (Holl, 2005) .66
Table 19: Analogy of the graphical notations of BPM and control flow modeling ...................66

6.5. Index of Figures

Figure 1: Example of the current modeling style (Sueddeutsche Zeitung, April 14, 2008 [online]) ....3
Figure 2: Methodology of my master thesis ...........................................................................10
Figure 3: Example of an unstructured EPC model (Erkollari et al, Epistemological foundations of requirements engineering, 2007, 35) .................................................................11
Figure 4: Girl and globe (Quibeldey-Cirkel, 1994:15) ..........................................................17
Figure 5: Typical example of the current BPM style in the form of a UML activity diagram 1 ....23
Figure 6: Typical example of the current BPM style in the form of a UML activity diagram 2 ....25
Figure 7: Improved business process model in the form of a UML diagram .......................26
Figure 8: Well structured business process model in the form of a UML diagram ...............27
Figure 9: Genealogical tree of diagramming notations ..........................................................30
Figure 10: Demonstration of the order process ......................................................................34
Figure 11: Demonstration of a flowchart diagram – Original notation .................................39
Figure 12: Strength and weaknesses profile of control flow charts .......................................40
Figure 13: Demonstration of a control flow chart – Improved notation ...............................41
Figure 14: Notation of an activity on arc network (Heinrich, 2003:222) .............................44
Figure 15: Demonstration of an activity on arc network ..........................................................45
Figure 16: Strengths and weaknesses profile of NMT activity on arc networks ....................46
Figure 17: Schema for activity on arc networks .................................................................46
Figure 18: Schema for activity on node networks ..................................................................47
Figure 19: Demonstration of switching operations in a P/T-Net (Gadatsch, 2005:79) ..........49
Figure 20: Demonstration of a Petri net (P/T) without marks – Original notation .................50
Figure 21: Strengths and weaknesses profile of Petri Nets ......................................................51
Figure 22: Demonstration of a Petri net (P/T) without marks – Improved notation ..........52
Figure 23: Demonstration of a Nasi Shneiderman diagram – Original notation ....................55
Figure 24: Strengths and weaknesses profile of Nasi Shneiderman diagrams .......................56
Figure 25: Demonstration of a Nasi Shneiderman diagram – Improved notation ...............56
Figure 26: IDEFO box and arrow graphics (KBSI - Knowledge Based Systems Inc., 2006, [Online]) ...58
Figure 27: Demonstration of an IDEF0 diagram .................................................................59
Figure 28: Demonstration of an IDEF3 diagram (Gadatsch, 2005:72, translated by KG) .........60
Figure 29: Strengths and weaknesses profile of IDEF0/SADT diagrams ...............................61
Figure 30: Symbols used in data flow diagrams (DeMarco, 1979:51) .....................................63
Figure 31: Demonstration of a data flow diagram - Original notation ................................64
Figure 32: Strengths and weaknesses profile of data flow diagrams ......................................65
Figure 33: Improvement suggestions for the control flow representation in DFDs ......................66
Figure 34: Demonstration of a data flow diagram - Improved notation .................................66
Figure 35: Demonstration of a swim lane diagram - Original notation ................................66
Figure 36: Strengths and weaknesses profile of swim lane diagrams ....................................66
Figure 37: Demonstration of a swim lane diagram - Improved notation ...............................66
Figure 38: Demonstration of a block diagram - Original not. (left) and Improved notation (right) .66
Figure 39: Strengths and weaknesses profile of block diagrams ........................................66
Figure 40: Demonstration of an EPC diagram - Original notation ......................................66
Figure 41: Strengths and weaknesses profile of EPC diagrams ...........................................66
Figure 42: Demonstration of an EPC diagram - Improved notation .....................................66
Figure 43: Demonstration of a PROMET diagram - Original notation ................................66
Figure 44: Strengths and weaknesses profile of PROMET diagrams ....................................66
Figure 45: Demonstration of a PROMET diagram - Improved notation ...............................66
Figure 46: Demonstration of an UML activity diagram - Original notation ..........................66
Figure 47: Strengths and weaknesses profile of UML activity diagrams ...............................66
Figure 48: Demonstration of an UML activity diagram - Improved notation .........................66
Figure 49: Demonstration of an extended event driven process chain (eEPC) - Original notation...66
Figure 50: Strengths and weaknesses profile of eEPC diagrams ..........................................66
Figure 51: Demonstration of an extended event driven process chain (eEPC) - Improved notation.66
Figure 52: Demonstration of a BPM diagram - Original notation .......................................66
Figure 53: Strengths and weaknesses profile of BP diagrams .............................................66
Figure 54: Demonstration of a BP diagram - Improved notation .........................................66
Figure 55: Demonstration of an object oriented EPC - Original notation .............................66
Figure 56: Strengths and weaknesses profile of oEPC diagrams .........................................66
Figure 57: Demonstration of an object oriented EPC - Improved notation ............................66
Figure 58: Structure of a process (Holl, 2000) .....................................................................66
Figure 59: Gestalt of a head of a horse (Holl, 2000) .........................................................66
Figure 60: Man’s face to a female body (Holl, 2000) ..........................................................66
Figure 61: Establishment of feature based boundaries (Riedl 1987:158f, translated by KG) ....66
Figure 62: Sub-processes shown as a mathematical step function (Holl, 2000, translated by KG) .66
Figure 63: Diagram of a process with sub-processes, events and features (Holl, 2000) ..........66