Mobile User Interfaces in Business Process Management

Concepts and Implementation of Mobile User Interfaces for a BPM-system with Focus on Usability

Prof. Dr. L. Nastansky

Wintersemester 2003/2004

Betreuer:
Dipl.-Inform. Stefan Smolnik

Vorgelegt von:
Susanne Glißmann

Wirtschaftsinformatik
Matrikel Nr.: 6019266

Moselstr.19
50859 Köln
Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

St. Gallen, den _____________________________  _____________________________
(Datum)  (Unterschrift)
# Table of Contents

1 INTRODUCTION ........................................................................................................1

1.1 MOTIVATION .................................................................................................1

1.2 OBJECTIVE .................................................................................................2

1.3 STRUCTURE .................................................................................................3

2 FUNDAMENTAL TERMS & THEMATIC CLASSIFICATION ......................................4

2.1 BUSINESS PROCESS MANAGEMENT ............................................................4

2.1.1 Business Process ......................................................................................4

2.1.1.1 Traditional Input-Process-Output Model ...................................................4

2.1.1.2 Holistic View of a Process .......................................................................6

2.1.2 Management of Business Processes ............................................................7

2.1.2.1 Competitive Advantage ..........................................................................7

2.1.2.2 BPM Approaches .....................................................................................9

2.1.3 BPM-Systems ...........................................................................................11

2.1.3.1 Requirements ..........................................................................................11

2.1.3.2 Core Capabilities .....................................................................................12

2.2 HUMAN-COMPUTER INTERACTION ................................................................14

2.2.1 User Interface ............................................................................................15

2.2.1.1 Definition ................................................................................................15

2.2.1.2 Interface Categories .................................................................................16

2.2.1.3 UI Models ................................................................................................17

2.2.2 Usability .....................................................................................................19

2.2.2.1 Definition ................................................................................................19

2.2.2.2 Usability Goals ........................................................................................20

2.2.2.3 Usability Heuristics ..................................................................................22

2.2.3 Human-Computer Interaction Design .........................................................26

2.2.3.1 Identify Needs & Establish Requirements ...............................................27

2.2.3.2 Design & Redesign ..................................................................................27

2.2.3.3 Prototypical Implementation .....................................................................28

2.2.3.4 Evaluate ....................................................................................................28

3 INTRODUCTION TO THE BPM-SYSTEM TRAXION ......................................29

3.1 FUNDAMENTAL CONCEPTS ..........................................................................29

3.1.1 BPM Lifecycle ..........................................................................................29

3.1.2 BP Definition ............................................................................................30

3.2 PROCESS EXECUTION MANAGER ..................................................................34

3.2.1 Interaction Operations .................................................................................34

3.2.2 System Architecture ....................................................................................36

3.2.2.1 Front-End ................................................................................................37

3.2.2.2 Back-End ................................................................................................38

3.2.3 Current Web User Interface ........................................................................39

3.2.3.1 Input and Output Devices ........................................................................40

3.2.3.2 Graphical Appearance .............................................................................40

4 CONCEPT OF A MOBILE USER INTERFACE ..................................................44

4.1 NEEDS AND REQUIREMENTS ........................................................................44

4.1.1 Identify Needs ............................................................................................44

4.1.1.1 Current Factors of Influence ..................................................................44
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1.2</td>
<td>M-Business as Future Factor of Influence</td>
<td>46</td>
</tr>
<tr>
<td>4.1.1.3</td>
<td>Need for a Mobile User Interface</td>
<td>48</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Interpret Requirements</td>
<td>50</td>
</tr>
<tr>
<td>4.1.2.1</td>
<td>User Requirements</td>
<td>51</td>
</tr>
<tr>
<td>4.1.2.2</td>
<td>Context of Use</td>
<td>51</td>
</tr>
<tr>
<td>4.1.2.3</td>
<td>Data Requirements</td>
<td>52</td>
</tr>
<tr>
<td>4.1.2.4</td>
<td>Functional Requirements</td>
<td>53</td>
</tr>
<tr>
<td>4.1.2.5</td>
<td>Usability Requirements</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>Conceptual Design</td>
<td>55</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Device Classification</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1.1</td>
<td>Cell Phone</td>
<td>57</td>
</tr>
<tr>
<td>4.2.1.2</td>
<td>Pager</td>
<td>58</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>PDA</td>
<td>59</td>
</tr>
<tr>
<td>4.2.1.4</td>
<td>Communicator</td>
<td>60</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Application Solutions</td>
<td>60</td>
</tr>
<tr>
<td>4.2.2.1</td>
<td>Thin Voice Client</td>
<td>61</td>
</tr>
<tr>
<td>4.2.2.2</td>
<td>Thin Data Client</td>
<td>62</td>
</tr>
<tr>
<td>4.2.2.3</td>
<td>Smart Client</td>
<td>63</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Future mobile User Interface</td>
<td>64</td>
</tr>
<tr>
<td>4.2.3.1</td>
<td>Advanced Handheld &amp; Smart Client</td>
<td>65</td>
</tr>
<tr>
<td>4.2.3.2</td>
<td>Evolution to a Thin-Smart Hybrid</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Physical Design</td>
<td>66</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Fundamentals</td>
<td>67</td>
</tr>
<tr>
<td>4.3.1.1</td>
<td>Online &amp; Offline Mode</td>
<td>67</td>
</tr>
<tr>
<td>4.3.1.2</td>
<td>Mobile Editor</td>
<td>68</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Task Data</td>
<td>68</td>
</tr>
<tr>
<td>4.3.2.1</td>
<td>Task Lists on the Server</td>
<td>69</td>
</tr>
<tr>
<td>4.3.2.2</td>
<td>Task Lists on the Client</td>
<td>70</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Screen and Navigation</td>
<td>73</td>
</tr>
<tr>
<td>4.3.3.1</td>
<td>General Screen Structure</td>
<td>73</td>
</tr>
<tr>
<td>4.3.3.2</td>
<td>Web UI Consistency</td>
<td>74</td>
</tr>
<tr>
<td>4.3.3.3</td>
<td>Screen Layout</td>
<td>75</td>
</tr>
<tr>
<td>4.3.3.4</td>
<td>Navigation Structure</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>Prototypical Implementation</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Platform Considerations</td>
<td>82</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Front-End Software Decisions</td>
<td>82</td>
</tr>
<tr>
<td>5.1.1.1</td>
<td>J2ME Application as Client</td>
<td>82</td>
</tr>
<tr>
<td>5.1.1.2</td>
<td>J2ME Record Store for the Data Storage</td>
<td>84</td>
</tr>
<tr>
<td>5.1.1.3</td>
<td>Palm OS Device as Mobile Device</td>
<td>85</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Java Servlet as Back-End Interface</td>
<td>86</td>
</tr>
<tr>
<td>5.2</td>
<td>Core Implementation</td>
<td>86</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Mobile Client’s Components</td>
<td>87</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Back-End Extension</td>
<td>92</td>
</tr>
<tr>
<td>5.3</td>
<td>Known Limitations</td>
<td>93</td>
</tr>
<tr>
<td>6</td>
<td>Outlook</td>
<td>94</td>
</tr>
<tr>
<td>6.1</td>
<td>Possibilities for Further Development</td>
<td>94</td>
</tr>
<tr>
<td>6.2</td>
<td>Deployment in Extended Scenarios</td>
<td>95</td>
</tr>
<tr>
<td>7</td>
<td>Summary</td>
<td>97</td>
</tr>
<tr>
<td>8</td>
<td>References</td>
<td>99</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>CONTENT</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>STRUCTURE &amp; UTILIZATION OF THE CD</td>
<td>A</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>VENDOR LIST</td>
<td>C</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 – Classification of Business Processes .......................................................... 5
Figure 2 – Business Process IPO Model .................................................................... 6
Figure 3 – BPM-System’s Core Capabilities .............................................................. 13
Figure 4 – UI Models ............................................................................................... 18
Figure 5 – Attributes of System Acceptability .......................................................... 19
Figure 6 – Simple HCI Lifecycle ............................................................................ 27
Figure 7 – TRAXION BPM Lifecycle ..................................................................... 29
Figure 8 – Task Instance’s States ............................................................................ 33
Figure 9 – Editor’s Task Lists ................................................................................ 35
Figure 10 – System Architecture .......................................................................... 37
Figure 11 – Web UI Screen: Task List .................................................................... 41
Figure 12 – Web UI Screen: Task ......................................................................... 42
Figure 13 – M-Business ......................................................................................... 48
Figure 14 – Process Optimization through a Mobile UI ......................................... 49
Figure 15 – Mobile Device Classification .............................................................. 56
Figure 16 – Handheld Classification ...................................................................... 57
Figure 17 – Application Architecture Spectrum ..................................................... 61
Figure 18 – Mobile Tasks ...................................................................................... 69
Figure 19 – Task List: All Tasks ............................................................................ 69
Figure 20 – Task Lists: Current Tasks & Completed Tasks ...................................... 71
Figure 21 – Task Instance Synchronization Conflicts .............................................. 72
Figure 22 – Mobile UI Screen Structure ................................................................. 73
Figure 23 – Web UI ↔ Mobile UI Conversion ........................................................ 74
Figure 24 – Mobile UI Screen: Main Menu ............................................................. 76
Figure 25 – Mobile UI Screen: Task List ............................................................... 77
Figure 26 – Mobile UI Screen Functions ............................................................... 78
Figure 27 – Mobile UI Screen: Task Details ........................................................... 79
Figure 28 – Mobile UI Screen: Task .................................................................... 79
Figure 29 – Abstract Navigation Structure ............................................................. 80
Figure 30 – Navigation Structure in Detail ............................................................... 81
Figure 31 – Extended System Architecture ............................................................ 82
Figure 32 – Component: Task Handler ................................................................. 90
Figure 33 – CD Interface .................................................................................... A
List of Tables

Table 1 – UML Meta-Model Layer 2: Metamodel ..............................................................30
Table 2 – UML Meta-Model Layer 3: Model.................................................................31
Table 3 – UML Meta-Model Layer 4: User Objects......................................................32
Table 4 – Software Products used in the Prototypical Implementation..........................C
Table 5 – Software Products mentioned in this Thesis..................................................D
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>Business Process</td>
</tr>
<tr>
<td>BPM</td>
<td>Business Process Management</td>
</tr>
<tr>
<td>BPR</td>
<td>Business Process Reengineering</td>
</tr>
<tr>
<td>BPMS</td>
<td>Business Process Management Suite</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Stylesheets</td>
</tr>
<tr>
<td>E-Business</td>
<td>Electronic Business</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>HCI</td>
<td>Human-Computer Interaction</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Server</td>
</tr>
<tr>
<td>IPO</td>
<td>Input-Process-Output</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ITRA</td>
<td>Input Transforming Resource – Active</td>
</tr>
<tr>
<td>ITRP</td>
<td>Input Transformed Resource – Passive</td>
</tr>
<tr>
<td>J2ME</td>
<td>Java 2 Platform, Micro Edition</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform, Enterprise Edition</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Pages</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PEM</td>
<td>Process Execution Manager</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Messaging Service</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>SyncML</td>
<td>Synchronization Markup Language</td>
</tr>
<tr>
<td>TQM</td>
<td>Total Quality Management</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

Limited time, fast movement and constant change determine the business world of the 21st century. The increasing number of companies entering the saturated markets causes a high level of competition. Thus, the customer becomes more and more the dominant force in the supplier-customer relationship. To retain existing customers, as well as to gain new customers, the company must stand out against its competitors by providing competitive advantages.

One essential approach for the achievement of a higher competitive advantage is the improvement and the automation of the company’s business processes. A complete new design or a redesign of a business process aims at its cost reduction, quality and cycle time improvement. Consequently, the execution of a well-designed process bears good prospects to deliver a final product or service for a less expensive price and higher quality in a shorter time. The product or service becomes more interesting to the customer and therefore increases the company’s competitive advantage. However, as the business process environment’s conditions continuously change, the process may no longer fit to the changed environment. Its efficiency and effectiveness will decrease over time. To maintain the process’s competitive advantage, the company is forced to invest in its’ business processes by constantly designing and optimally managing them.

Deriving from the business process’s nature, this continuous procedure is immensely extensive and complex. Thus, it is appropriate to apply specific software referred to as Business Process Management-systems (BPM-systems). BPM-systems consist of several software components providing features to enable the design, execution, management, analysis and optimization of a broad range of business processes. The process execution component will be the specific focus of this thesis.

The execution component applies the defined business process logic to actual process instances and manages the input-output flow of the process. Employees of various industry sectors and occupational fields work with this type of software to perform and complete their work specific tasks. For this, the software component provides a PC user interface (UI) that allows them to access its back-end application. Such an IT support for the execution of business processes enables major efficiency improvements since a consistent UI is provided in multiple process situations.

However, since the software deals with a broad range of business processes including
1 Introduction

various different user environments, a single PC-based UI actually limits the applicability of such a business process management software to certain types of processes. The activities in these processes must be executed with a PC at hand. A recent business trend, which promises further growth potentials and access to new markets and business fields is mobile business or mobile commerce. A common challenge of mobilizing existing services and applications is the limited functionality of mobile devices, especially as far as user interface components are concerned.

Thus, to allow the usage of business process management-systems in processes in mobile commerce or mobile business, a new software component must be designed with special regard to Usability, a field of the discipline of Human-Computer Interaction (HCI). The following questions arise in this context:

- Which user types require a mobile user interface or can at least profit from it and what are the specific requirements?
- Which components of BPM software systems are used by these users?
- How must the mobile user interface be designed in order to achieve maximum usability and efficiency in the broadest possible range of user environments?

1.2 Objective

Objective of this thesis is to answer the questions posed in the previous paragraph. The basis for the analysis is the BPM-system TRAXION provided by CommerceQuest, Inc. The theoretical background is mainly based on the scientific and practical state of the art in the field of Business Process Management (BPM) as well as the field of Human-Computer Interaction (HCI).

The results are as follows:

- The definition of characteristics of users for whom a mobile user interface to BPM-systems is appropriate
- The specification of typical requirements of these users
- The definition of a structured method to design and implement mobile user interfaces, particularly for BPM-systems
- A prototypical implementation as a proof of concept of the described method, based on sample hardware, but portable to other platforms
1 Introduction

1.3 Structure

The subsequent chapter introduces the fundamental terms and thematic classification that underlies this thesis. At first, the methodology of Business Process Management is explained. On the basis of the given business process definition, companies’ objectives to manage their business processes, as well as current BPM approaches are presented. To provide a good understanding of the BPM-system TRAXION, the common nature of a BPM-system is illustrated. The second major topic deals with the discipline of HCI. From this discipline, the field of usability and the field of human-computer interaction design are taken to provide guidelines for the assurance of a user interface that is well accepted and used by the end user.

The third chapter conveys fundamentals of the BPM lifecycle and the BP definition specific for TRAXION. Furthermore, it describes TRAXION’s software component, the Process Execution Manager (PEM), for which an additional UI will be designed. PEM’s user interaction operations, the system architecture and the current user interface are explained to create a foundation for the further design process.

According to HCI’s guidelines, the fourth chapter proceeds in two phases. The first phase analyzes the current needs of PEM’s current and potential future end users. Afterwards, for the discovered need for a mobile UI, the requirements are pointed out. In the second phase, the mobile UI is designed in two steps. The conceptual design aims at finding the best

The fifth chapter depicts the next phase of the human-computer interaction design, the prototypical implementation of the mobile UI concept. Here, platform considerations for the front-end, as well as for the back-end are made. In addition, the design of the mobile UI’s inner structure is illustrated.

The sixth chapter shows possibilities for the further development of PEM’s mobile UI, as well as its deployment in extended scenarios.

In conclusion, the seventh chapter summarizes the results of this thesis.

The appendix describes the structure and utilization of the CD attached to this thesis. Furthermore, it comprises two lists of software; one which has been evaluated for usage in the prototype and the other which has been actually used.
2 Fundamental Terms & Thematic Classification

Fundamentals of this thesis are two major thematic fields. One focal point is the methodology of Business Process Management (BPM) with the concentration on BPM-systems. The other emphasis lies on the discipline of Human-Computer Interaction (HCI) and the presentation of its goals and practical methods.

2.1 Business Process Management

Business Process Management deals with the management of business processes, whereby the management of these processes is performed by BPM-systems.

2.1.1 Business Process

In the following, the traditional and the current enhanced process view are introduced.

2.1.1.1 Traditional Input-Process-Output Model

“A process”, according to Thomas H. Davenport, “is .. a specific ordering of work activities across time and place, with a beginning, an end and clearly identified inputs and outputs …”¹. Inputs are broken into two categories: active and passive². The active input is referred to as input transforming resources (ITRA) defining the resources that transform other resources. The input transformed resources (ITRP) are the passive input. The described process definition is the traditional input-process-output (IPO) model³.

Moreover, this IPO model represents a system. “A system is a collection of interrelated parts which form some whole”⁴. It is categorized either as a closed system – when it is completely self-supporting, or as an open system – when it depends on the environmental input and the distribution of its output. Therefore, a process is a system that is open and is, furthermore, “put into operation to produce a specified result”⁵.

Another essential feature of a process is the composition of a hierarchy of sub-processes⁶. To reduce the complexity of large processes and to allow re-use, processes can be broken down into sub-processes. These sub-processes can be continuously decomposed until the smallest units, the activities, are reached. All units of this major process are based on the IPO model. However, the difference between activity and process is the difference between

¹ Davenport 1993, p.5
² Barnes 2001, p.18
³ Pall 1999, pp.60
⁴ Barnes 2001, p. 5
⁵ Pall 1999, p. 60
⁶ Jacka et al. 2001, pp.23
2  Fundamental Terms & Thematic Classification

part and whole\(^7\). An activity is a unit of work in general assigned to one single input transforming resource. It only generates a fraction of a result. In contrast, a process is a related group of multiple activities or sub-processes. This group is allocated to several input transforming resources that create a result.

Within a system, the processes, sub-processes and activities are cross-functional\(^8\). They are separated by their boundaries and interact with each other under certain rules through their input-output interfaces. A process uses the output from a prior process as input. This input is transformed into new output that is then again given to the next process. The size of a process, the number of activities and their degree of interaction differ between the various systems.

*Business Process*

Surveying the system organization as a process, it is composed of numerous sub-processes\(^10\). The number differs between industry sectors and organizations.

These sub-processes are categorized into three broad types: core processes, support processes and management processes\(^11\). Core processes focus on the external customer. They are initiated by customer requests and fulfill customer needs.\(^12\) They directly add value to the business. To ensure this, core processes rely on support processes. Support processes concentrate on internal customers. They only add direct value by providing a suitable working environment. Management processes administer core and support processes. Furthermore, they concern themselves with planning on the business level. Typical business processes include product development, customer order fulfillment, and financial asset management.

The IPO model is clearly transparent in these processes. A core process is used for demonstration purposes. Here, input triggered by a customer’s commission is received by

---

7 Harrington et al. 1997, p. 1-3  
8 Barnes 2001, p 5-6  
9 Ould 1995, p.26  
10 Harmon 2002, p.20-21; Jacka et al. 2001, p.27  
11 Ould 1995, p.2-3  
12 Ould 1995, p.26
2 Fundamental Terms & Thematic Classification

the business enterprise. Adding value to this input generates the output that is then discharged into the environment. The ITRP in an organization are for example materials, information, customers’ knowledge and energy. The ITRA are facilities and staff. The result of this process is in an ideal case customer satisfaction accomplished by the provision of needed goods and services to the customer. This example can easily be carried over to support and management processes.

![Business Process IPO Model](image)

Figure 2 – Business Process IPO Model

The previously described organization processes are defined as business processes. The following comprising definition can be made: “A business process is a logical set of customer-supplier relationships that drive work activities performed by people and other resource-consuming assets in order to produce committed results at specified points in time”. In this thesis, the business process is either referred to as process or business process (BP).

2.1.1.2 Holistic View of a Process

Although, the traditional IPO model is seminal and has been used in companies for many years, it is not applicable to the entire range of BPs. The IPO model only comprises static processes with definable inputs and outputs, interacting with a stable environment. But, as Scheer in his new approach to business processes points out, in companies exist “a range between two extremes: (1) completely predictable, repeatable procedures with a pre-determined workflow and no defined commitments and associated accountabilities; and (2) entirely probabilistic, often ad hoc processes depending on well-defined but changeable commitments and accountabilities”. In order to apply one process model to the entire process range, Scheer developed a holistic process view that incorporates the traditional

---

13 Pall 1999, p.162-163
14 Ould 1999, p.2
15 Pall 1999, p.68
16 Pall 1999, p.75-82
17 Pall 1999, p.179
IPO model.

In this enhanced model, he focuses on the information flow, the humans in a process and the dialogues between them\textsuperscript{18}. His idea is that in addition to the work activities managed in a process, the commitments also have to be managed. Therefore, he designed the framework of the customer-supplier dialog, whereby customer and supplier in a process are either external or internal. Over a certain timeframe, the two players in a business process, customer and supplier, communicate with each other. The dialog can be categorized into four sequential phases that are representatives of business transactions: definition phase, negotiation phase, performance phase and assessment phase.

Every process starts with the definition phase that is either a request from the customer or an offer from the supplier. In the subsequent phase, the negotiation phase, the supplier and the customer determine the conditions or change the requirements. Afterwards, in the performance phase, also called work phase, the supplier produces the requested products or services and commits them to the customer. Finally, the assessment phase completes the dialog by the customer’s acceptance.

This holistic process view implies a high dynamic in a process environment. Although this model mainly focuses on the information, the environment’s new characteristic also influences transformed and transforming inputs. This derives from the changed conditions conveyed by the unpredictable information input. The result is a high adjustment demand in the transformation process, consequently in the entire process\textsuperscript{19}. The holistic model is applicable to the complete range of processes. Consequently, the static processes, their structures and behaviors are no longer stable\textsuperscript{20}.

2.1.2 Management of Business Processes

The previous chapter emphasized the diversity and complexity of business processes. In the following, the need for the management of these processes is explained. Furthermore, the different BPM approaches are presented.

2.1.2.1 Competitive Advantage

In 1903, Henry Ford introduced a new approach in automobile manufacturing. It was characterized by the automobile development as a single process, whereby, the sequential

\textsuperscript{18} Pall 1999, p.171-180
\textsuperscript{19} Barnes 2001, p.5
\textsuperscript{20} Smith et al. 2003, p.46
order of each activity ensured greater profit in the production\textsuperscript{21}. This approach already showed the business world how important it is to improve the way a company makes business.

Since then, companies’ awareness of their process-centered business view has increased. They deviated from the so-called silo thinking that was characterized by major concentration on each activity, and rather focused on the entire process. For this, they began analyzing, improving and managing their business processes. This development slowly occurred in the last century but really progressed in the last decades for the following reasons\textsuperscript{22}.

With increasing competition, power of the customer and influence of the government\textsuperscript{23}, the companies were forced to find ways of surviving in the current market by creating competitive advantage. Competitive advantage can be achieved, according to Michael E. Porter, through cost leadership, differentiation or niche specialization. The sophisticated deployment of BP plays a decisive role and is a key factor in gaining and maintaining competitive advantage. For example, to provide less expensive goods or services, the organization must use low price inputs or must have more efficient processes than its rivals. A transformation of input into output always is combined with adding value to the company. The amount of value that is added differs from process to process and depends on the efficiency of the respective process\textsuperscript{24}. Therefore, a more efficient process adds more value to the company and leads to competitive advantage. However, companies that copy successful business processes and use them for their own purposes, threaten this competitive advantage. To counteract this behavior, Porter advocates that companies focus on an entire network of interacting efficient business processes that are created to follow the strategic organizational goals. The high interactivity and the relationships between these processes complicate the copying of single processes. In summary, competitive advantage is achieved by creating efficient processes and, in recent years, has been guaranteed in the long term through a highly interactive process network\textsuperscript{25}.

Nowadays, maintaining competitive advantage is even more complicated since business processes are further exposed to the fast, unstable and unpredictable environment. Therefore, they must possess intrinsic efficiency and be efficient in their interaction with

\begin{flushleft}
\textsuperscript{21} Harmon 2002, p.5  \\
\textsuperscript{22} Harmon 2002, p.23  \\
\textsuperscript{23} Pall, pp.17  \\
\textsuperscript{24} Barnes 2001, pp.33  \\
\textsuperscript{25} Barnes 2001, pp.50; Harmon 2002, pp. 21
\end{flushleft}
other processes. Furthermore, they must remain efficient in this current environment. This can be achieved by high adaptability and agility within the process. In addition, to gain high competitive advantage, the business process view awareness must not be limited to core processes but also must be extended to the entire range of business processes.

With the goal of gaining and maintaining competitive advantage, companies have been strongly focused on their business processes. For this, they have applied business process improvement and management theories, practical methods and technologies.

### 2.1.2.2 BPM Approaches

Although companies have always analyzed and refined processes, the first concrete business process improvement ideas, methods and theories emerged as recently as in the early 1980s and developed throughout the 1990s until now. And even now, they are still in progress and changing. The following paragraphs give a short overview of the key BPM ideas in recent years.

*Total Quality Management (TQM)* is an empirical management approach that was the standard during the 1980's. It is applied to improve the quality of products and services using a totally new approach to quality assurance. In TQM, quality is definable and measurable. Objectives and requirements are formulated to be achievable. Specific projects are implemented to ensure that quality is improved where that is a necessity. Hereby, basis quality methods are always addressed to customers’ requirements. Although TQM has been very successful, it also exhibited weaknesses. For example, the component of continuity was missing and there was a limited view of the overall system performance. Therefore, TQM has not been applicable as a complete BPM method but more as a basis for quality assurance that has been integrated in many other BPM approaches.

The advent of *Workflow Automation* and *Enterprise Resource Planning (ERP)* strongly focused on automating existing processes and replacing departmentally focused legacy systems with new software modules that were designed to work together. Business process automation is performed by workflow systems that control the flow of documents from one employee to another. Higher process efficiency arises from process cycle time reduction and automated process control. However, the pure automation and management of processes does not completely harness all possible process improvement opportunities. For example, process bottlenecks or unnecessary activities are not automatically eliminated...

---

26 Evans 2001, chapter 1; Smith et al. 2003, p 67
27 Harmon 2002, p.36
28 Pall 1999, p.22-23
through this approach. However, the deployment of ERP software enables the integration of departments and functions of a company. For this, a single software program is used that accesses one database. ERP software provides standard processes for the organizational process flow. With this standardization a certain general process quality is assured. But since, these standards do not absolutely match every BP, deployments of ERP software often require adjustments in the organization and the process flow\textsuperscript{29}.

Business Process Reengineering (BPR), also known as business process innovation or new process design, is the most radical approach of all and started in the early 1990s. The idea behind BPR is to take a fresh look at the organization, completely ignoring the present processes, and defining new processes using BPR methods. One powerful instrument to achieve a very high degree of optimization is to apply new information technology (IT). Furthermore, BPR theorists like Thomas Davenport, Michael Hammer and James Champy emphasize the improvement of the entire process and not only sub-processes. The large size of BPR projects often implies extensive redesign efforts. It is therefore the most costly and time-consuming approach with the highest degree of risk associated with it. There is no doubt that BPR implies great success in some cases, but in other cases it can cause tremendous harm to the organization\textsuperscript{30}

Another approach is Continuous Process Improvement. This effort focuses on the constant improvement of sub-processes, activities, and tasks and is often performed after a process has been improved by BPR. The constant improvement is reasonable since a company’s environment is in a state of constant change. After an improvement approach, the process may rapidly lose its actuality. A process that was once efficient may no longer be up-to-date. It becomes necessary to adapt the environment in order to regain the same efficiency\textsuperscript{31}.

Business Process Redesign is the contrary BPM approach to BPR. Instead of creating completely new processes, the existing processes are redesigned. The prevailing organizational structures and process flows are examined for efficiency. Afterwards improvement opportunities are defined and used in the subsequent redesign. Some goals of process redesign include cycle time reduction, error elimination, automation, standardization and simplification\textsuperscript{32}.

\textsuperscript{29}Harmon 2002, p.27-30; Smith et al. 2003, p.29-33
\textsuperscript{31}Harrington et al. 1997, p.13-14
\textsuperscript{32}Harrington et al. 1997, p.8-9
2 Fundamental Terms & Thematic Classification

A new technology approach that has been applied increasingly in the aforementioned BPM approaches is Internet enabling of business processes. Business processes performable through the Internet lift the restrictions to Intranets and organizational networks. They leverage a company’s partners and customers to become integrated into the automated and managed corporate business processes. Furthermore, they allow employees to perform their work activities remotely by using the Internet. This manner of conducting business on the Internet is defined as Electronic Business (E-Business)\textsuperscript{33}.

2.1.3 BPM-Systems

In previous years, the methodology of BPM and the offered BPM software mainly focused on the initial design of processes with assurance of their improvement and on process automation. Nowadays, this goal has been enhanced and adjusted to the dynamic nature of business processes with the absence of prediction. With the aid of BPM-systems the process will not only be designed but also highly maintained during its lifetime. The following chapter gives a short introduction to BPM-systems. See e.g. Smith et al. 2003, p.71-98 for a more extensive analysis.

2.1.3.1 Requirements

BPM-systems are deployed to fulfill the need of rapid changes in business processes. For this, the process must be continuously observed and analyzed. When changes in the BP’s environment appear, adjustments become necessary and the business process must be changed rapidly - ideally in real-time\textsuperscript{34}. Therefore, the entire BPM lifecycle must be managed and controlled by the system to remain a continuously adjusting process. The management should not be exclusively restricted to the management of the executed processes. It should rather be enhanced to the management of the process design\textsuperscript{35}.

Moreover, rapid change is only guaranteed if all people involved in a process are actively integrated in the entire process management lifecycle. In particular, the business people must be able to influence the design process since they are the first people who recognize the need for a change. Thus, it is important to abstract the business process from the implementation technology\textsuperscript{36}. Adequate process views must be provided to the variant process members including the manager, the employee, the business analyst and the programmer enabling a top-down, as well as bottom-up process modeling. It is contrary to

\textsuperscript{33} Smith et al. 2003, p.28-29
\textsuperscript{34} Pall 1999, pp 76
\textsuperscript{35} Burlton 2001, p.81-125; Smith et al. 2003, p.76-77
\textsuperscript{36} Ficher 2004, pp.301
the previous BPM approaches where business analysts designed the process and technicians implemented the process. Furthermore, enabling all variant members and striving for collaborative reengineering, the process design is affected by multiple completely diverse influencing factors\textsuperscript{37}.

For the process design, it is essential that the information and the organization of a process are inexorably connected. Whereby, the process controls the consumption, generation and transformation of information. Information access by organizational members is defined through access and ownership permissions. Process activities are explicitly allocated to employees\textsuperscript{38}.

Moreover, process management should not be limited to the intrinsic process design characteristics. It should also facilitate the integration of other applications and back-end systems into the process. Here, a fast and automated adjustment of the integrated applications will also improve BPM-systems adaptability\textsuperscript{39}.

Finally, BPM-systems’ mandatory business capability must allow businesses to take control of their current and future business needs\textsuperscript{40}.

2.1.3.2 Core Capabilities

The previously illustrated requirements lead to the following capabilities of BPM-systems: Discovery, Design, Deployment, Execution, Monitoring, Interaction, Control and Analysis. As can be seen in figure 3, they are arranged in a BPM lifecycle striving for constant business process improvement. This BPM lifecycle can be seen as a continuous process\textsuperscript{41}.

- **Discovery**

When a business process is first entered into a BPM-system, all participants of this process, including humans, machines and applications, must be identified. Additionally, their interaction must be captured as it is essential to gain a good understanding of the event flow, information flow and control flow. This overall process knowledge will enable the detection of improvement possibilities in the subsequent design phase.

\textsuperscript{37} Smith et al. 2003, p.88
\textsuperscript{38} Ficher 2004, p.299
\textsuperscript{39} Smith et al. 2003, p.251-252
\textsuperscript{40} Smith et al. 2003, p.94
\textsuperscript{41} For further explanations to the BPM lifecycle and its core capabilities see Smith et al. 2003, p.98-94
2  Fundamental Terms & Thematic Classification

- **Design**
  The capability of process design in a BPM-system is achieved through graphical process mapping and modeling. The design is not limited only to the design of new processes but includes the manipulation and redesign of already existing processes. The goal of design is to optimize process activities, rules, members, interactions and relationships.

![Figure 3 – BPM-System’s Core Capabilities](image)

- **Deployment**
  Good system support of the deployment phase is achieved through a largely automated procedure. Hereby, the process is rolled out to the various process participants.

- **Execution**
  After a process is distributed to all its members, the system ensures that it is also carried out correctly. The BPM-system must manage the task flow, generate process data and store this data permanently. According to the defined responsibilities of a task, the task is always assigned to the appropriate entity.

- **Interaction**
  For the interaction with the process, the BPM-system provides process desktops and process portals. The power of these interaction interfaces is enhanced when combined with the Internet. The interaction possibilities include work assignment, task management and data entry. Furthermore, users are able to observe and monitor processes.

---

42 Smith et al. 2003, p.90
2 Fundamental Terms & Thematic Classification

- Monitoring & Control
The monitoring and control of a process is either performed manually through interaction or automatically by the system. For a process to run efficiently and remain error-free, unexpected errors and exceptions must be detected. When an error occurs, process interventions are permitted. For example, during execution it must be possible to change the assignment of tasks and input transforming inputs.

- Analysis
The process requires constant improvement whereby its conditions change rapidly. Therefore, to be able to maintain its efficiency and to further increase it, the process must be analyzed. Process analysis happens on past data as well as simulated process predictions. On the one hand, the performance of the completed process is examined, and on the other hand, the intended process is simulated using “what-if” validations.

- Optimizations
Based on the analysis, bottlenecks, deadlocks and other inconsistencies are detected and will be eliminated either automatically by the system or manually by a process participant. The optimization activity merges with the redesign of a process.

2.2 Human-Computer Interaction
BPM-systems are powerful tools to optimize, automate and manage business processes. In particular, the underlying technologies and applied logic provide companies with great chances to improve their businesses. BPM-systems consist of strong back-end tools interacting with other applications or machines. When using a BPM-system, the user interface (UI) is the part of the system that the user perceives, and therefore, it plays an essential role in the overall success of specific software. The reason for this is that the human user is the decisive factor in the human-computer interaction. If the communication between the user and the computer fails or is unsatisfying, the user will decide against the software despite all opportunities the application has to offer. Therefore, it is important to understand the human-computer interface. The gained knowledge must be used for the creation of an efficient communication channel for humans and applications. For this purpose Human-Computer Interaction (HCI) discipline is applied. It is “…devoted to helping people meet their needs and goals by making computing technology accessible,
2 Fundamental Terms & Thematic Classification

meaningful, and satisfying. Thus, it deals with the user interface, its quality and its design process.

2.2.1 User Interface

Although user interface (UI) – also called human-computer interface, human-machine interface or simply interface - is a widespread term, the majority of the average computer users cannot give a clear definition. This is explainable by most users’ way of looking at the computer and their evolving computer literacy. These users only understand one side of the human-computer interaction. This side represents the human world and the computer components that are visible to the user. But, they are unaware of the other side that consists of the invisible computer parts. Furthermore, this limited comprehension of a UI is strengthened if users mainly come into contact with good UIs. The reason for this derives from the main principle of user interface design: A human-computer interface is a good interface if it is transparent to the user.

For the development of a UI, it is essential to gain a good understanding of user interfaces, to learn about available UI categories, as well as the variant prevailing UI perspectives.

2.2.1.1 Definition

Raskin defines a human interface as follows: “The way that you accomplish tasks with a product – what you do and how it responds – that’s the interface.” “It is “… humane if it is responsive to human needs and considerate of human frailties.” This general definition of a user interface can be applied to the wide range of diverse human interfaces. It covers for example, remote controls, coffee machines, watches, phones, etc. However, it does not define explicitly the specific human-computer interface.

The human-computer interface is simply the parts of a computer and its software that the user sees, hears, touches or talks to. It is the combination of things that allow the communication between user and computer. Everything that a user tells the computer is input and everything that the user receives is output. The way this happens entirely depends on the user interface. A communication between two humans is characterized by an information exchange possible without any translation aids. In contrast, the communication between a human and a computer requires interaction devices. To be able

43 Carroll 2001, p.191
44 Hewett et. al 1992, p.12-14
45 IBM 2004, p.4; Berkun 1999
46 Raskin 2000, p.2
47 Raskin 2000, p.6
to talk to the computer, the user utilizes input devices. These input devices include computer keyboards, joysticks, microphones, mice, pens, touch-sensitive screens, etc. For the communication from computer to user, output devices are utilized. Output devices are hardware components such as displays, headphones, printers or speakers. Nowadays, a wide range of various interaction devices exists. The right selection of the adequate interface device for the system depends on the user, his environment and the system’s nature. Each device is optimized for particular situations and is less efficient for others. Finally, the third part of the UI is represented by the controls. These are the software elements the user perceives and sees on the display when using the application. Menus, pushbuttons, radio buttons, sliders belong to these UI components.\footnote{IBM 2004, p.4-5; Shneiderman 1997, pp.305}

### 2.2.1.2 Interface Categories

Since the first computer was developed, computer users have communicated with computers through interfaces. In the early days most computer users had a good technological understanding and often wrote the software for themselves. Thus, the first user interfaces were targeted at the efficiency of those users. With the growing diffusion of computer applications to users with fewer or no technological skills, the user interfaces had to become more sophisticated. Higher sophistication of user interfaces in return allowed for a further diffusion to even less technologically skilled users. This development is clearly visible in the evolved user interface categories.\footnote{Berkun 2000 a; Rosson et al. 2001, p.9}

The first UI was the \textit{Command-Line-Oriented Interface}. Its only form of information exchange was text - or rather text commands - that the user sent to the computer. The applied input device was the keyboard. The output device was solely the desktop. Although, the course of communication was interactive and dynamic, the dominant part in this early human-computer interaction was the computer. The user was the one using the computer to perform his tasks, but it was the computer that set the style, the rules and the language of their communication. Incorrect commands called by the user resulted in failures, misunderstandings, communication breakdowns and computer rejections. Therefore, an improvement of the human-computer communication channel became necessary.\footnote{Preece et al.2002, p.42-44; Shneiderman 1997, p.72-73; Shneiderman 1997, pp.275}

The first step in this direction was the introduction of the \textit{Menu-Oriented UI}. Here, the UI is improved by the provision of selectable command lists. Consequently, the user is not
2  Fundamental Terms & Thematic Classification

forced to know everything about the system. The UI functions in a more supportive manner by giving suggestions to the user. Whereby, the user only needs to recognize the needed command from the list\textsuperscript{51}.

With the visual design of the *Graphical User Interface (GUI)* not only the communication style, but also the appearance and behavior of the interface adjusted to the human world. The application of real elements - such as e.g. the recycle bin or document folders - in combination with the direct manipulation of these elements create a simplified human world inside the computer. Moreover, an improvement of the UI has been achieved through uniform graphical elements in the various applications. Finally, the user communication has been intensified and developed by the ability of users to work with several applications at a time\textsuperscript{52}.

Whereas the previous UI adjusted to the user by copying physical elements into the computer, the *Future UI* has left the traditional PC environment restricted by desktop, keyboard and mouse. It is characterized by “...ubiquitous, invisible, embedded, tangible, virtual, active, integrated, interconnected, interoperable, and mobile”\textsuperscript{53} capabilities. Nowadays, a wide range of these kinds of UI exists, and a further increase will be expected in the following years\textsuperscript{54}. However, one example for these UI is a mobile user interface in combination with a mobile device such as a handheld. It will be explained in further detail in Chapter four.

**2.2.1.3 UI Models**

When defining a user interface, people essentially make use of a user interface model. This model serves for description and examination purposes of a UI’s complex nature. Depending on the speaker’s perspective on the UI, as well as his computer literacy a different model is applied. In general, there are three types of UI models: the programmer’s implementation model, the user’s mental or so-called conceptual model, and the designer’s manifest model. The following illustrations refer to Preece et al. 2002, p.55; Cooper 1995, p.27-44; IBM 2004 b.

\textsuperscript{51} Shneiderman 1997, pp.235; Preece et al.2002, p.44-47
\textsuperscript{52} Cooper 1995, pp.193; Nielsen 1994, p.57-61; Preece et al. 2002, 41-49
\textsuperscript{53} Carroll 2001, p.195
\textsuperscript{54} Carroll 2001, p.221-223
2 Fundamental Terms & Thematic Classification

Figure 4 – UI Models\textsuperscript{55}

- Implementation Model

The implementation model is “the actual model how a device works…”\textsuperscript{56}, “conveying a complete and unambiguous design to the programmers”\textsuperscript{57}. This model defines the implementation details and describes system internals. The programmer applies this model for the development of the system.

- Mental Model

Since the user doesn’t need to know the complex UI details, he forms a mental shorthand for his understanding - the mental model. The mental model represents a mental image that each user intuitively creates when he interacts with the system. It comes into being from putting together sets of perceived rules and patterns in a way that gives a good explanation of the situation. Although it represents what a user thinks and the reasons for it, it does not necessarily reflect the situation and its components accurately. Furthermore, while interacting with the system, the user continuously increases his knowledge. With this learning process, the user’s conceptual model may be expanded. Since users have different expectations and understanding of a system, the conceptual model also differs for each user.

- Manifest Model

The third model describes “the interface components and relationships intended to be seen by users and .. to become part of each user’s conceptual model …”\textsuperscript{58}. It helps, amongst other things, the user to understand the system and to inform the programmer about users’ requirements. Therefore, the manifest model is placed somewhere in between the

\textsuperscript{55} Cooper 1995, p.27
\textsuperscript{56} Cooper 1995, p.27
\textsuperscript{57} IBM 2004 b
\textsuperscript{58} IBM 2004 b
implementation and the user model. Depending on the proximity to the implementation model – if the manifest model “… follows the reality …”\footnote{Cooper 1995, p.29} - or the mental model – if the model follows “… user’s imagination”\footnote{Cooper 1995, p.29} - the manifest model will create either a difficult or an easily understandable UI for the user. This derives from the likely mental model that can take much of the complexity out of a UI, resulting in the UI’s simplification. “The ability to tailor the manifest model is a powerful lever that the software designer can use positively or negatively”\footnote{Cooper 1995, p.30}.

### 2.2.2 Usability

For the design of a user interface it is essential to ensure that it will be accepted and used by the user. This objective is the topic of usability. It is explained in the following.

#### 2.2.2.1 Definition

Usability is the narrow concern compared to the larger issue of system acceptability - the overall acceptance of a computer system\footnote{Nielsen 1994, p.24-26}.

![Figure 5 – Attributes of System Acceptability\footnote{Nielsen 1994, p.25}](image)

System acceptability consists of social and practical acceptability. If the user accepts a system’s purpose the social acceptability is achieved. In that case the practical acceptability has to be further analyzed. Practical acceptability is divided into the following categories: cost, compatibility, reliability, usefulness, etc.\footnote{Nielsen 1994, 24}... Whereby, “usefulness is the issue of whether the system can be used to achieve some desired goal”\footnote{Nielsen 1994, 24}.

It is subcategorized into Utility and Usability. Utility examines the system’s availability of “... the right kind of functionality that enables users to carry out all their tasks in the way...”\footnote{Nielsen 1994, 24}.
they want to do them. However, Usability is the question of how well users can this functionality. The user and his or her communication with the system’s user interface is the focus of attention.

In the beginning of user interface design, UIs were created with the goal of being user-friendly. But because of the diversity of humans, this general principle didn’t result in an overall user-friendliness applicable for each user. Therefore, goals have been created taking this problem into consideration. Reaching for these goals enables the optimization of a system’s human-computer interaction for all its defined users.

### 2.2.2.2 Usability Goals

Nielsen defines five usability goals: learnability, efficiency of use, memorability, few and noncatastrophic errors and subjective satisfaction. Preece further enhances these goals by a sixth attribute: effectiveness. Moreover, she also expands the goals errors and satisfaction. First of all, she includes the goal few and non-catastrophic errors in the more comprehensive term safety. In addition to this, she replaces satisfaction with the term user experience goals in which satisfaction can be found as one of its attributes. These goals apply to all aspects of a system with which a human might interact, including deployment, installation and maintenance procedures. They are used to help design and measure a system and its user interface. In the following, usability goals will be explained. See Nielsen 1994, p. 23-37, as well as Preece et al. 2002, p.13-20 for a more extensive analysis.

- **Learnability**

Starting to use a system, the user has to learn how it works. The initial ease of learning depends on its degree of difficulty and the user’s previous knowledge of other systems. A computer expert will probably understand a system faster than a computer novice. Although both learning curves start out with the user being able to do nothing, the course of the expert’s learning curve will be more rapid and he will reach his steady state of proficiency faster than the novice. In general, people don’t like spending a long time learning how to use a system. Therefore, the system needs to be easy to learn, and the time it takes to achieve a sufficient level of proficiency should be short. Learnability is an essential objective for the measurement of a UI’s usability since the first experience most
2 Fundamental Terms & Thematic Classification

people have with a new UI is that of learning how to use it.\textsuperscript{67}

- **Efficiency**

When the learning curve flattens out and the steady state level of performance has been reached, the user knows how to use the system. At this point the efficiency of a system can be measured. Efficiency appraises a system’s supportiveness. It quantifies how a system helps users carrying out their tasks and if it can sustain a high level of productivity. Very often the goal of efficiency comes into conflict with the goal of learnability. This derives from the fact that a system, which is easy to learn and explanatory, is consequently slower and therefore less efficient. One solution for this conflict problem is to make the system adjustable to variant proficiency levels, providing different stages of supportive interaction.\textsuperscript{68}

- **Effectiveness.**

A more general goal of the defined usability goals is effectiveness. Effectiveness deals with the question of whether the system does what it is supposed to do. It also examines its capability and how good it is at doing what it should do. It deals with considerations such as if the system allows the people to learn well, carry out their tasks efficiently and access the information they need. Finally, another concern is for example, if it is possible to use the system when it is needed\textsuperscript{69}.

- **Memorability**

This goal defines the system’s nature to be easy to remember. The casual user should be able to return to the system after some period of not having used it without relearning it. A user remembers the usage of the system based on his previous learning. For example, good memorability can be achieved if all things important for the system’s use are made visible on the screen. Therefore, the user doesn’t need to retain as much information about the system as they would with a system with limited visibility. This design goal is especially important for infrequently used systems\textsuperscript{70}.

- **Safety**

Another essential usability goal is the safety of a system. It describes its external and internal safety, “… protecting the user from dangerous conditions and undesirable

\textsuperscript{67} Nielsen 1994, p.27-30; Preece et al. 2002 p.16  
\textsuperscript{68} Nielsen 1994, p. 30; Preece et al. 2002 p.14  
\textsuperscript{69} Preece et al. 2002 p.14  
\textsuperscript{70} Preece et al. 2002 p.17
2 Fundamental Terms & Thematic Classification

situations”71. External safety refers to the external conditions where the user works, and represents the ergonomic aspect. One example is the remote access to machines in hazardous environments. The second understanding of safety defines the safety within a system. A secure system prevents the user “… from making errors by reducing the risk of wrong keys or buttons being mistakenly activated…”72. Different errors occur in a system, some are easy to correct and others are irreparable errors. Where an error occurs, the system should provide the user with various means of recovery. A safe interactive system ensures that the user becomes confident and encourages exploration and trying out new operations73.

• User Experience

The final usability goals are user experience goals, applied in order to find out if the user enjoys the system. These goals identify the user’s subjective opinion of the system by examining attributes such as satisfaction, pleasure, fun, entertainment, help, motivation, aesthetical pleasure, supportiveness of creativity, rewarding experience, emotional fulfillment74.

On closer examination, as seen earlier in the learnability-efficiency conflict, there never exist a single correct answer as to how a usable system can be created. Trade-offs arise and the designer always must weigh the advantages and disadvantages resulting from the set usability goal combination. Achieving simultaneously optimal scores for all usability attributes is often impossible and several alternatives usually exist. In the end, the designer chooses - in his opinion - the best combination of all usability goals. In addition to the conflicts within usability goals, usability goals also come into conflict with complementary goals of a system’s overall acceptability75.

2.2.2.3 Usability Heuristics

To achieve usability goals, the designer makes use of usability heuristics. Heuristics are “a rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood”76. In usability, heuristics are categorized into two principles: design principles – “as reminders of what to provide and

---

71 Preece et al. 2002, p.14  
72 Preece et al. 2002, p.15  
73 Nielsen 1994, p. 32-33  
74 Preece et al. 2002, p.19  
76 Hyperdictionary 2004
what to avoid when designing an interface"\textsuperscript{77} – and usability principles “assessing the acceptability of interfaces, used during heuristic evaluation”\textsuperscript{78}. Design principles and usability principles often merge and it is difficult to differentiate between them.

In current literature, numerous usability heuristics exist. Although their names and classification might be different, they can be seen as ten well accepted usability heuristic rules distinguishable in most heuristics. These rules are explained in the following paragraphs.

• \textit{Simplicity}

This is a major heuristic rule for user interface design. A good user interface always keeps the UI simple and straightforward, striving for a natural dialog with the user. One applied technique is prioritizing usability over function\textsuperscript{79}. This does not mean that a “… lack of functionality…” occurs, but “… a fast initial learning curve and consideration for the number of concepts the user needs to understand”\textsuperscript{80} will be achieved. It counteracts the received opinion “… that the more features a product has, the better it will be”\textsuperscript{81}. The tenet of \textit{less is more} should always question the necessity of each object, information contents of the screen, the choice of feature interactions and interaction mechanisms. A usable system presents exactly the information the user needs at exactly the time and place where it is needed. Nowadays, even the simple tasks become immensely complicated, therefore it is more than ever essential to keep the simple simple\textsuperscript{82}.

• \textit{Consistency}

Another important heuristic rule is consistency. This rule seeks for standards in UIs: similar operations and similar elements achieving similar tasks and is one of the most basic heuristic rules. Consistency produces an increase in user’s confidence using the system. Non-existence of consistency makes it difficult to understand and remember the system, and therefore makes the user more prone to mistakes\textsuperscript{83}.

• \textit{Minimization of User Memory Load}

In addition to this, it is important to convey as much as possible mental work from the human to the computer. This results in a minimization of user memory load, the third

\textsuperscript{77} Preece et al. 2002, p.28
\textsuperscript{78} Preece et al. 2002, p.28
\textsuperscript{79} IBM 2004 c
\textsuperscript{80} Berkun 1999
\textsuperscript{81} Berkun 1999
\textsuperscript{82} Nielsen 1994, p.115-123, IBM 2004 c
heuristic rule. Here, visibility of elements, objects and functions play a decisive part. For a human it is always easier to see and recognize an object instead of recalling it from the memory. This derives from the fact that a human’s passive vocabulary is often larger than his active vocabulary. Using a menu oriented interface, and not a command-line interface is one example of how to reduce the user’s memory load84.

• **Speak the Users’ Language**

Viewing interactions from the user’s perspective and reproducing the human physical environment within the computer helps the user to work with the computer more naturally. For this purpose, user interface metaphors are a possible way to achieve a mapping between the computer system and some reference system known to the users in the real world. Furthermore, it is expedient to make use of affordance. Affordance enables people to know intuitively the behavior or meaning of an object. This is achieved through the association of this object with other things85. For example, by throwing a document into a computer’s recycle bin, it is clear to the user that the document is deleted, or will be completely deleted when the recycle bin has been emptied. Furthermore, the computer should speak in a language that is easy to understand for the user. The designer should create a user interface in the user’s native language and not in a foreign language. In addition to this, a user interface should also avoid unusual and incomprehensible words. By applying these techniques, the computer makes it easier for the user to work with it86.

• **Feedback**

Feedback is another essential heuristic goal. The computer should always inform the user about relevant data such as what the computer is doing, how it interprets the user’s input, and what action has been accomplished. This data is not only restricted to error messages, but also includes confirmation or delay messages. Here, it is fundamental, that the response time – the time it takes the computer to send the feedback – is as fast as possible. If the system has long response times and the computer cannot provide fairly immediate response, continuous feedback should be provided to the user using a progress bar indicator for example87.

• **Preventing Errors**

Although, no system is safe from errors, the number of errors should always be minimized.

---

84 Nielsen 1994, p.129-132
85 Carroll 2001, p.51-66; Preece et al. 2002 p.25
This can be achieved by preventing errors. The system prevents errors by the assistance of the user interaction with the computer. One example is the provision of correct inputs that the user can choose from. This already avoids error situations in the first place. Another way of minimizing sources of errors is input validation. Moreover, errors can be avoided by clearing the system from multiple modes. “A mode is a state the program can enter where the effects of a user’s action changes from the norm – essentially a behavioral detour” 88. In some cases, changing from one mode into another results in confusion and frustration of the user as he has to adjust to the new mode situation. Errors may therefore occur. Unfortunately, modes are almost impossible to avoid totally in an interface of some complexity89.

- **Error messages**

If an error occurs, the system should present an adequate error message, helping the user to understand and recover from the error. According to Shneiderman, the error message should follow four rules. An error message should be phrased in clear language and avoid obscure codes; it should be precise rather than vague; it should constructively help the user solve the problem; and it should be polite and should not intimidate the user or put the blame explicitly on the user90.

- **Clearly Marked Exits**

The user feels unsure in situations from which he cannot escape. Therefore, the system should always provide the user with ways to cancel a started function or exit the system. Here, it is essential that these exits are extremely visible in the interface. This derives from the fact that the user wishes for a simple escape from a problem - especially in situations where he is inexperienced and doesn’t feel comfortable91.

- **Shortcuts**

To enrich the user interface with more flexibility and efficiency of use, the designer employs so-called shortcuts. Shortcuts are user interaction accelerators such as function keys, command keys, objects that can be double-clicked and buttons in a dialog. Furthermore, shortcuts include pen computers, virtual realities and some mouse interfaces. Shortcuts are used to perform frequently used operations faster92.

---

88 Cooper 1995, p.69  
89 Shneiderman 1997, p.76-77  
90 Shneiderman 1997, p.373-380  
91 Nielsen 1994, p.138-139  
92 Nielsen 1994, p.139-142
2 Fundamental Terms & Thematic Classification

- **Help and Documentation**

  The last heuristic rule to mention is the provision of help and documentation to the user. Although, a designer strives for the creation of a system where help or documentation is not necessary, this is often impossible. Reasons for this include the trade-offs of the usability goals described earlier. Therefore, a good system should possess helpful documentation of the system. This documentation can exist as printed offline documents or be integrated in the system. In particular, context-sensitive help facilitates the user in finding the needed information\(^\text{93}\).

2.2.3 Human-Computer Interaction Design

According to Jennifer Preece, “design is a practical and creative activity, the ultimate intent of which is to develop a product that helps its users achieve their goals”\(^\text{94}\). Human-computer interface design is the multi-disciplinary theory and the practice of designing interactive products that are usable to the user. Therefore, the philosophy of user-focused design takes center stage during the entire design process. Every consideration is made and every decision is reached with the future user in mind.

The design process of a system is often seen as lifecycle model of this product. In software engineering literature, there are several different design lifecycles, comprising the established waterfall lifecycle, spiral lifecycle model, Rapid Applications Development (RAD), and so on\(^\text{95}\). In addition to those, there also exist specific lifecycle models in HCI, inheriting from software engineering lifecycles and appending a stronger focus on Usability. One of these is Preece’s Simple HCI Lifecycle\(^\text{96}\). It has been chosen to exemplify as the basis for the design and the prototypical implementation of a new user interface. As every other usability lifecycle, the simple lifecycle model focuses on users, specific usability and user experience goals\(^\text{97}\); as well as iteration. It is initiated if a new product invention must be created, or if an already existing system has to be adjusted. In the end, it results in a final product. As can be seen in figure 6, the simple lifecycle model consists of four interacting and iterating activities: (1) Identify needs and establish requirements; (2) (Re-) design; (3) Prototypical implementation; and (4) Evaluate. In the following, these phases are described.

---


\(^{94}\) Preece et al. 2002, p. 165

\(^{95}\) Preece et al. 2002, p. 165-168

\(^{96}\) Interaction Design, p. 187

\(^{97}\) Preece et al.2002, p.182-200
2.2.3.1 Identify Needs & Establish Requirements

To develop an effective system, the system must match the distinct requirements interpreted from the users’ needs\(^9\). It is not always given that people know their needs, since they are not informed about all existing possibilities. Instead, the designer approaches the users needs by understanding the user’s characteristics and capabilities.

For this, first of all, the users must be defined. Very often the users can be described by allocating them to various characteristic categories. Characteristics include their computer knowledge, their age and their job skills. When choosing characteristics, it is essential to stay open minded since human characteristics that do not seem to be important at first sight, may decisively influence the use of a UI. Beside a good user definition, the tasks these users perform also have to be analyzed for their frequency, criticality, time to complete, difficulty, and division of responsibility.

Using the obtained information, the designer detects the system’s requirements. Two main kinds of requirements exist in usability design: functional and non-functional requirements. Functional requirements say what the system should do. Non-functional describe the system’s constrictions. Non-functional requirements are further sub-categorized into: data requirements; environmental requirements or context of use. Whereby, context of use comprises the physical, social requirements, as well as the technical environment, user requirements and usability requirements. Finally, it is to say that requirements always emerge from the users, their tasks and their environment or the context they occur in.

2.2.3.2 Design & Redesign

Once the requirements are determined, the design phase begins. The design phase is

\(^8\) Preece et al.2002, p.186  
\(^9\) Preece et al.2002, p.201-236
subdivided into two sequential activities: conceptual and physical design. Conceptual design describes what the UI will do and how it will behave, while physical design considers the detail of the product such as the colors, sounds, images to use, menu design, and icon design. It is essential, that in this phase several design ideas are suggested and compared to each other. A good User Interface is achieved by a high user involvement during this phase\textsuperscript{100}.

### 2.2.3.3 Prototypical Implementation

To evaluate the design ideas, it is reasonable to produce interactive versions of the alternative design ideas. These interactive versions are termed prototypes. A prototype can be anything from a paper-based storyboard to a complex document. It is applied for interaction with an envisioned product to gain some experience of using the prototype in a realistic setting, as well as to explore imagined uses. For this, two different kinds of prototypes exist, the low-fidelity and high-fidelity prototype. A low-fidelity prototype is one that does not look very much like the final product such as storyboarding and sketching. It is useful because it tends to be simple, cheap and quick to produce. However, a high-fidelity prototype resembles the final product or uses materials that would be expected to be in the final product. It is rather time and cost intensive to create but it gives more detailed information about the later product deployment. Sometimes a prototype is even further enhanced to a final product\textsuperscript{101}.

### 2.2.3.4 Evaluate

In the following evaluation phase, the prototype is surveyed and tested under examination of the defined requirements and usability goals. The results are used to improve the existing system. A multitude of methods and techniques are applicable in this phase. Since this thesis is not concerned with the evaluation of the implemented prototype, it will not be further discussed.

---

\textsuperscript{100} Preece et al. 2002, p.239-278

\textsuperscript{101} Preece et al. 2002, p.239-278
3 Introduction to the BPM-System TRAXION

The BPM-system that underlies this thesis is the Business Process Management Suite (BPMS) TRAXION of the company CommerceQuest Inc. CommerceQuest Inc. was founded in 1991 and is headquartered in Tampa, Florida, USA. It specializes in the development of BPM software102.

In the following, TRAXION is explained to conveying a general comprehension that is essential for the further reading. It presents the system’s fundamental concepts, as well as the software components that have been relevant for this project. In addition, its current web user interface is analyzed.

3.1 Fundamental Concepts

The next two chapters give a brief introduction to TRAXION specific features comprising its realizations of its BPM and BP definitions.

3.1.1 BPM Lifecycle

As already mentioned earlier, business process management is a continuous process constantly striving for the optimal design and execution of efficient and flexible business processes. Consequently, TRAXION has been designed following this BPM lifecycle. The TRAXION BPM lifecycle consists of three sequential, iterating phases that are composed of BPM-system’s core capabilities103. For each phase, TRAXION provides the required software components.

In the first phase, Analyze & Design, processes and organization structures are either newly designed or redesigned. The analysis is conducted by the examination of performed processes and simulation. The next phase of Execution & Integration automates the designed objects. It contains all tools needed for the management and coordination of enterprise assets, people, systems and resources. One tool that is applied in this phase is the Process Execution Manager (PEM). It is the software tool for which this thesis’s concept of a mobile UI is designed, and therefore, will be

---

102 www.CommerceQuest.com
103 Chapter 2.1.3.2
3 Introduction to the BPM-System TRAXION

explained in further detail later on. The third BPM phase is Management and Optimization. In this phase, the executed processes are monitored and examined for improvement opportunities.

3.1.2 BP Definition

TRAXION’s IPO model is defined as follows. Input Transforming Resources (ITRA) are humans or machines. Input Transformed Resources (ITRP) is information. The transformation process is the consumption, generation and transformation of TRAXION’s ITRP by TRAXION’s ITRA. The business process type that is realizable in TRAXION is located in the middle part of Scheer’s defined process range. It is a predictable and repeatable procedure with a pre-determined workflow. Although it is well-defined, commitments and accountabilities may change. Core processes, support processes, as well as management processes represent TRAXION’s business processes.

In TRAXION, a business process is defined according to the three layers metamodel, model and user objects of the Four-Layer UML Meta-Model Architecture.

- Layer 2: Metamodel

The Metamodel defines the elements that are necessary to create a process definition in the following layer.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: Metamodel</td>
<td>process element; sub-process element; task element; task field element; TRAXION-task field element; customized-task field element; link element; supervisor; editor; reader;…</td>
</tr>
</tbody>
</table>

Table 1 – UML Meta-Model Layer 2: Metamodel

It provides a task element (TRAXION’s process activity), as well as a sub-process element. Furthermore, the provision of a link element enables the connection of task elements and sub-process elements to form a network that is defined as a process element. This process element represents TRAXION’s IPO metamodel. It can also be used as a sub-process element in another process element. However, task attributes further specify the task element. They consists of elements e.g. for the allocation of process members (humans or

---

104 Chapter 2.1.1
105 CommerceQuest 2003 a
106 OMG 2003, p.63: (1) Meta-Metamodel: Infrastructure for a metamodeling architecture; (2) Metamodel: Language definition for the specification of a model; (3) Model: Language definition for the description of domain information; (4) User Objects: Definition of a specific information domain
In TRAXION, process members consist of supervisor, editor and reader elements. The supervisor is defined as the owner of a process and its entire tasks. He possesses process control and task management capabilities in the event of a delay or dissonance. The member editor possesses the rights to process and complete a task. The final member is the reader of a task. He has exclusive read accessibility.

Task field elements consist of customized-task field elements, as well as TRAXION-task field elements. The first group of field elements is specific to each process definition, and therefore, must always be newly defined by the process designer in the following layer. The second group of task fields applies to all process definitions. These TRAIXON-task field elements include e.g. task name, task scheduled start date, task scheduled end date and task duration time.

- **Layer 3: Model**

A concrete composition of the metamodel’s elements forms the process definition.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
</table>
| 3: Model | • *Process Definition*: Insurance customer claim  
| | • *Task Definitions*:  
| | - Customer’s notification of a damage  
| | - Damage inspection by the insurance adjuster  
| | • *Link Definitions*:  
| | … ➔ Customer’s notification ➔ Damage inspection ➔ …  

**Specification of task definition** *Damage inspection*  
- *Editor Definition*: Workgroup: Insurance Adjuster  
- *TRAXION-Task Field Definitions*:  
  - *(Task Name*: Damage inspection); *(Duration Time*: 1 day); …  
- *Customized-Task Field Definitions*:  
  - *(Name*: date of damage; *Type*: date); *(Name*: kind of damage; *Type*: String); *(Name:.. ; *Type:..);…*  

Table 2 – UML Meta-Model Layer 3: Model  

A process definition is created in the design phase of a BP and it is unique in TRAXION’s
3 Introduction to the BPM-System TRAXION

environment. In this phase, tasks are determined and connected through link elements. Customized-task field elements must be further defined by specializing their field names and field types. In addition, some of TRAXION-task fields are further defined in this phase. For example, the task definition’s name and duration time are set.

Furthermore, the allocation of possible ITRA to process member elements (reader, editor and supervisor) defines the explicit members of the process definition. Possible ITRA are taken from company’s organizational entities and carried over into TRAXION. For the integration, TRAXION applies common organization structure theories\(^{107}\). For this, TRAXION adopts and slightly adjusts these organization structures to its own needs identifying four different entities: **specific resources, departments, workgroups** and **roles** (synonymous with units). Resources are persons or materials and present the smallest units of the organization. They are characterized by possession of skills and memberships of other entities. Departments are persistent and made up of one or more persons. In general, an organization holds a hierarchy of departments and their sub-departments. In addition, workgroups exist. Workgroups consist of multiple persons, but their nature is more temporary, as they will be created for specific projects. The final entity is a role. A role, in contrast to the others, is a more abstract entity. Roles are applied to achieve more flexibility into the organization structure. This flexibility derives from the fact, that roles are used as representatives of specific functions or activities in an organization. A role is a placeholder for an unknown person.

- **Layer 4: User Objects**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>4: User Objects</td>
<td>• <em>Process Instance</em>: Insurance customer’s claim no. 1254</td>
</tr>
<tr>
<td></td>
<td>• <em>Task Instance</em>: Damage inspection case no. 1254</td>
</tr>
<tr>
<td></td>
<td>• <em>Editor Instance</em>: Workgroup: Insurance Adjuster: John Smith</td>
</tr>
<tr>
<td></td>
<td>• <em>Task Field Instances</em>:</td>
</tr>
<tr>
<td></td>
<td>(Date of Damage: 03/05/2004); (Type of Damage: Flooding);</td>
</tr>
<tr>
<td></td>
<td>(Task Activation Date: 03/06/2004); …</td>
</tr>
</tbody>
</table>

Table 3 – UML Meta-Model Layer 4: User Objects

When a process is started, this model layer is entered and a concrete process instance of a process definition will be created. In general, several process instances to one process

\(^{107}\) Fischer et al. 2002, p.283-292
3 Introduction to the BPM-System TRAXION

definition will be instantiated over time. Characteristic for this model layer is its dynamic. During a process instance’s lifetime, information is processed and the process instance’s task instances change their states.

The information that is processed derives, on the one hand, from the customized-task field instances, and on the other hand from TRAXION-task field instances. Information stored in customized-task field instances is referred to as task data, and information stored in TRAXION-task field instances are referred to as task details.

When a new task instance is initialized, it goes through several stages of its lifecycle.

![Figure 8 – Task Instance’s States](image)

The completion of a prior task instance triggers the next task instance. This task instance is set into its initial state - the activated state. It characterizes the task instance by its visibility to the supervisor, all possible editors and the readers. From this state the task instance is lead over to the second state, the claimed state. This happens, when a potential editor claims the task instance. The editor is then the actual editor, responsible for the completion of the task instance. He is able to edit and save the task data, as well as to release and complete the task instance. Each action other than edit and save sets the task instance in a different state. Every other entity allocated to this task instance still sees the task instance, but nobody else but the supervisor is able to claim a task instance in this state. A task instance terminates and triggers the next instance when the actual editor has completed the claimed task instance. The task instance is then in its third state, the completed state. An optimal task instance performance rests in each state exactly once. But it happens that resetting the task instance into a previous state reverses the lifecycle. For example, this happens in the claimed state, when the actual editor releases its claimed task instance and puts it back into the activated state. Another example is the undoing of an already completed task. Then, the state changes from the completed to the claimed state. A task instance can only be undone, if the subsequent task instance has not been started.
3.2 Process Execution Manager

The Process Execution Manager (PEM) is one of TRAXION’s executing software tools. It is applied to ensure BPM lifecycle’s core capabilities execution, monitoring, interaction and control. As BPM runtime execution engine, it independently drives and manages the automation of business processes. It therefore embodies the core of TRAXION’s entire components. The PEM is deployed in B2B, enabling E-Business through its web user interface. It links human workflow, applications, and extends beyond enterprise boundaries.

By providing interaction capability to the mere end user, the businessman, high usability plays a major role in the system’s acceptability. With the advent of an additional user interface, CommerceQuest expects to address a wider user range. This is explained by higher system effectiveness that derives from the provision of multiple user interfaces. Consequently, with the rise of possible users, PEM will be able to extend its business process application fields.

PEM’s user interaction operations and architecture will be described in the next paragraphs. Furthermore, to be able to create consistency between current and intended user interface, the current web UI will be explained.

3.2.1 Interaction Operations


- Authentication & Authorization

PEM is endowed with secure authentication and authorization mechanisms. Every user of the system is forced to login with a user name and password. Using these two attributes, the system ensures only approved users the access to corporate data. Furthermore, the unique user name allows the system to provide each user with its intrinsic data, its allocated task instances and functions.

- Task Handling

The main feature that PEM offers is Task Handling. The system prepares each user of the user type editor or reader several lists of task instances. They are represented by My Tasks,
All Tasks and Undo Tasks and differ in their task instances’ user responsibility, state and functionality.

My Tasks includes all tasks instances for which the user possesses the rights of the actual or potential editor, as well as the reader. Partly, the task instances are also accessible to other editors holding the same rights. Furthermore, they are in the state activated or claimed. The functions, to these states are claim and release. When the user changes from the task list view to the task view of one task instance, the permissible functions extend to edit, save and complete.

All Tasks refers to all tasks instances that the user is permitted to see encompassing task instances in all different task instance states. With reference to the task instance state and the user’s responsibility for the specific task instance, different functions are provided in the task list and the task view.

Undo Tasks shows the completed tasks instances and allows the user to undo the task instances completed by him.

- Process Management

Process management is an operation that is subdivided into its two categories: process management by the user and process management by the system. Process management by the system is not further explained since it happens entirely in PEM’s back-end system and not in its user interface.

Process Management by the user allows the supervisor to manage the process instances manually. The user is supported by the system that informs him about delayed and critical task instances. In the event of an error in a process instance, the supervisor claims the error
causing task instance. Even if an editor has already claimed the task instance, the supervisor holds the right of claiming the same task instance. Afterwards, a task instance delegation by the supervisor to a different person is possible.

- **Process Monitor**

Process Monitor acts as a control function. The system compares the process instances and prepares process analysis for the supervisor. The user applies these reports as the basis for making decisions in the executed process. Furthermore, he can use it as basis for the redesign of the process definition.

- **Process Administration**

Process Administration is, primarily, handled by PEM’s system administrator. When a new process definition has been designed, it is saved as an XML-file. By uploading this XML-file into the PEM, the administrator imports a new process definition.

If errors occur during import of the new process definition or during the execution of a process instance, the errors can be viewed in the PEM.

- **Organization Administration & Management**

After creating a new organization in the Process Resource Modeler, an organization is imported, similarly to the import mechanism of a process.

### 3.2.2 System Architecture

The Process Execution Manager (PEM) is a typical web application using the World Wide Web (WWW) to provide application access. It consists, on the one hand of a strong and complex back-end application, and on the other hand, a very slim front-end with little independent application logic. Both, back-end on the server side and the front-end on the user side, are tightly connected to each other. On closer inspection, one sees that the back-end and front-end are composed of four interacting components: the user interface, the web server, the application server and the relational database system. These four components build the so-called four-tier-architecture, made up of presentation, web, application and data layer.\(^{109}\) In the following, PEM’s architecture is explained to the extent that it will enable the reader to easily follow the integration of the mobile user interface as well as the architectural requirements that the original architecture implies.

3.2.2.1 Front-End

To fully understand the large back-end, one must first comprehend the nature of the smaller front-end. The front-end is a so-called web browser interface or web UI. The user applies this kind of UI to access the application with aid of a web browser such as Internet Explorer or Netscape Navigator. Therefore, the application runs on every computer – e.g. personal computer, laptop, tablet PC - that supports web browsers, as well as a permanent Internet connection. The communication between this web UI and the back-end is ensured through *HyperText Transfer Protocol* (HTTP)\(^{110}\).

PEM’s UI is deployed as follows: The user launches the PEM by entering its specific Uniform Resource Locater (URL) into the web browser. With this URL the web browser knows the location of the application and starts a new user-computer communication session. The kind of dialog used is the HTTP Request-Response mechanism *Request Forwarding*. It “allows a component to send a request to one URL in an application and have it processed by a component at a different URL in the same application”\(^{111}\). The frequented resources of the URL include, among other things, Java Server Pages (JSP), which will be explained later on in this chapter. The content of PEM’s response is a HyperText Markup Language (HTML) file that will be interpreted by the web browser and displayed to the user. HTML comprises static data such as HTTP parameters, but also information about the presentation style and specific application data. Within an HTML file, the user sends a request to the application, receiving the next required HTML file afterwards.

---

\(^{110}\) W3C 1999  
\(^{111}\) Sun 2003
3.2.2.2 Back-End

The web browser interface serves as a data and file exchange instrument between PEM and the user. Apart from its ability to graphically prepare an HTML file for the user, it does not possess independent application logic. Consequently, the entire application logic is placed in PEM’s back-end.

- **Web Server**

A web server ensures PEM’s back-end connection to the Internet. This web server is located between application server and web browser. Therefore, in the previously described request response mechanism, the web server serves as receiving office for user’s requests, and furthermore, forwards these requests to the application server. Moreover, it acts as a provider for the application’s responses, sending the generated HTML files to the web browser. The web server also manages and controls web sessions. Common web servers are Microsoft Internet Information Server (IIS) and Apache. They are applicable for the support of the PEM application.

- **Application Server**

Application servers offer sharing and processing of application logic. Moreover, they are a connection to other back-end systems such as databases, legacy systems, third-party software and middleware, as well as core applications. With their interface developing mechanism and the techniques to apply applications to the web, they can perfectly be deployed for PEM’s Business Process Management engine.\(^{112}\) This BPM engine runs on IBM WebSphere vers.5 and Tomcat 4.1.27 of the Apache Software Foundation.

The **BPM engine** located on the application server is the core of PEM’s components since it comprises its entire specific application logic. The engine’s parts work together as follows: Java Server Pages (JSPs) are the application’s primary entrance to the outside world. The application server, therefore, uses them to interact with the application. Within the BPM engine, they are linked to cascading stylesheets and the BPM engine class library.

Although the JSPs are located on the back-end application and mainly possess application logic, they also strongly overlap with the PEM’s specific web browser UI. This has to be taken into consideration when a new additional UI is designed. Thus, it is important to gain a good understanding of JSP.

*A Java Server Page (JSP)* “…is a text-based document that contains two types of text:

\(^{112}\) Linthicum 1999, p. 131
static template data, which can be expressed in any text-based format, such as HTML; and JSP elements, which construct dynamic content.” The JSP describes how to process a request, as well as how to build a response. Furthermore, they endue the ability for accessing server-side objects. The PEM engine avails these characteristics by using JSP to control the navigation and the application flow. In addition to this, it uses their proficiency for obtaining presentation style information and application data. With this knowledge, the application server is enabled to create HTML files - the screens of the web browser UI.

The JSPs retrieve web browser UI’s presentation style from *Cascading Stylesheets* (CSS), which are also placed on the BPM engine. Stylesheets are used as “… a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents.” Therefore, they are only usable for this kind of UI - the web browser UI.

In addition to stylesheets, JSPs also call the methods of the *BPM engine class library*. They do this to make use of TRAXION’s own BPM features and logic by integrating them into the UI. The BPM engine provides, among others, following methods: operation and controlling of processes and tasks; management of organizational structures; authorization and authentication; preparation of data; and data access. The BPM engine class library is implemented in Java 2 Platform, Enterprise Edition (J2EE). J2EE “defines the standard for developing component-based multitier enterprise applications”.

### 3.2.1.1 Database

The final component in this four-tier-architecture is the database system. This database consists of an extensive database structure. It contains static and dynamic data. The database is used to store PEM application specific data, static process definitions as well as dynamic process content. Furthermore, it holds information about each state of the task instances and also the organizational structure and its members. The database is accessible to the developer only through data access functions in the BPM class library. PEM applied as databases DB2 or Microsoft SQLServer.

### 3.2.3 Current Web User Interface

Developing an additive user interface for a system, the designer must not only gain an established knowledge about its architecture, but also about the way the existing UI
interacts with its’ back-end application. Nevertheless, there is more than this that must be taken into consideration. Since the application represents an integrative unity, every other UI needs to be matched to this existing unity. This is achieved through the conservation of consistency between the original and the intended UI. Consequently, the developer has to learn more about the current UI: its input and output devices, and its graphical appearance. Further perceptions about the UI are attained by the examination of the UI’s presentation with regards to usability heuristics and usability goals. This approach will facilitate the developer in his future UI design process.

3.2.3.1 Input and Output Devices

Using a web browser interface such as PEM’s UI, implies the input and output devices that are characteristic for the personal computer (PC).

Consequently, users mainly apply the desktop or flat screen as the output device. Although, the Web UI is able to run on specific Personal Digital Assistants (PDA), it is very questionable to do this. This derives from the fact, that PEM’s original UI is primarily developed for large screens. Therefore, its usability on small screens cannot be assured. Keyboard and mouse act as PEM’s input devices. On the one hand, the keyboard enables the user to enter an immense amount of textual task data in a fast and very efficient way. On the other hand, the mouse provides the unknown user a simple mechanism to communicate with the application through direct manipulation of its screen elements.

In summary, the combination of desktop, keyboard and mouse is highly practical for PEM’s deployment in office environments.

3.2.3.2 Graphical Appearance

In the following, a brief introduction to PEM’s web browser UI’s main window structure is given. Afterwards the applied usability heuristics of this UI are analyzed.

PEM’s presentation structure follows the web navigation conventions that have “…mostly adapted from existing print conventions.”\textsuperscript{117} The web UI is, therefore, split up into three divisions: sections – comprising primary navigation (1) and subsections (2); content (3); and utilities (4).\textsuperscript{118}

\textsuperscript{117} Krug 2000, p.60
\textsuperscript{118} Krug 2000, p.65
PEM’s primary navigation (1) is created for the three different user types: end user, supervisor and administrator. Whereby, the end users are the editor or reader, that are defined within a process; the supervisor is the supervisor of a process; and the administrator presents the administrator of the application.

The user specific functionality is accessed by addressing subsections(2). PEM’s entire set of features, described in a previous chapter, is accessible through these subsections. By clicking a subsection’s tab, the appropriate function is made accessible and its content will be displayed in the content section. In the example above, figure 11, the end user made use of the feature Task Handling. It is located under primary navigation tab My Tasks and the subsection My Tasks. All the tasks for which the user is defined as reader or possible editor are now displayed in a task list. This task list presents task details including the process title, task name, process id, activation date, team - possible editor group, claimed by, scheduled start time and due date of the single task.

The user, then, either further filters these tasks by certain criteria, or he performs a task instance. Within this task list view, he is able to claim, release or print a task (3). Furthermore, the user can open and edit a task, by accessing its hyperlink. This action updates the content in the content section, displaying the screen in figure 12.

The utilities section (4), in general, provides information about how to use a side; it can also give information about its publisher. These are mainly typical elements that does not
fit in the before explained other sections. In the PEM the utilities section is used to access the system help and to exit the application.

**Examination of Usability Heuristics**

The screenshots in figure 11 and figure 12 clearly show the applied usability heuristics.

Both screens are characterized through their *simplicity*. When starting the application, the user is directly lead to his task list. Since, this task list claims nearly the entire screen, the user is presented with a good overview of his available tasks. The navigation and the functions remain in the background on the upper and lower screen.

However, *consistency* within the web browser is achieved through the similar screen structure in the various screens. The main structure described above is maintained in every screen. Furthermore, the structure in the content area also indicates consistency. The available functions for the specific task maintain their constant location independent of task or task list screen.

---

119 Krug 2000, p.65
3 Introduction to the BPM-System TRAXION

The large size of the desktop screen already implies the ability to minimize user memory load. Everything that needs to be visible for the user is made visible. In particular, the division of the navigation section in one primary and one sub section enables the user to foresee the possible navigation. Therefore, he or she must not remember or explore the way through the navigation.

The usage of affordance, by applying icons carried over from the user’s physical world follows the usability heuristic to speak the user’s language. Furthermore, the language that is actually used is characterized by its simplicity and clarity. Therefore, it is simple to understand for the user.

As can be seen in the upper part of the task screen in figure 12, a successfully performed action always provides feedback to the user. Here, the user has claimed a task and has been informed about the result.

In case of an error, instead of a positive feedback an error message would have been provided at the same place. Since this user information is integrated in the screen and not displayed in an extra dialog box, it does not disrupt his work but informs him in an adequate way.

The web browser UI prevents errors, on the one hand, by the data entry controlled by the system, and on the other hand, by not permitting incorrect data entries.

Clearly Marked Exits are noticeable in the main structure in the utilities. There, the user finds the logoff command. In addition to this, the user exits a screen by instantiating a new screen. Within a task screen, the close button is clearly visible.

The web browser UI does not support specific Shortcuts. It simply uses the well-accepted general Windows shortcuts such as e.g. Ctrl+c to copy text.

The web browser is kept understandable and simple to use, therefore Help and Documentation does not necessarily need to be integrated in this UI. The user may access help via an extra help file stored on the PC.
4 Concept of a Mobile User Interface

In the second chapter, the reader was introduced to the theoretical framework of Business Process Management and Human-Computer Interaction. In the third chapter, CommerceQuest’s Process Execution Manager (PEM) was presented. However, this chapter takes one step forward targeting the realization of PEM’s new user interface and the assurance of its usability. For this, the design process followed the HCI design life cycle reaching for usability goals and applying usability heuristics.

4.1 Needs and Requirements

As every other company, CommerceQuest always aims to maintain its competitive advantage. To retain and extend its market share, CommerceQuest constantly strives for the improvement of its products. With PEM’s enhancement through an additional UI the increase of BPM Suite’s system acceptability is intended since the provision of multiple UIs extends the number of possible end users. To determine the potential new end users, it is essential to learn more about PEM’s current, as well as future client’s employees and their needs.

4.1.1 Identify Needs

The examination of PEM’s user needs is conducted on the processes and organization structures already implemented by CommerceQuest. Further conclusions derive from studying current literature about mobile business.

4.1.1.1 Current Factors of Influence

- *Who are the Current Users?*

PEM’s user communities cover a wide range of different kinds of users. These communities are distinguishable by certain criteria, but they also often intimately intertwine.

The first kind of user group is the PEM specific system user consisting of the editor, the supervisor and the administrator. These users only make use of their specific system features and ignore the rest of the application. For example, the editor will never perform an administrative function.

---

120 Chapter 2.1.2.1
121 Chapter 2.2.3.1
123 Chapter 3.1.2
4 Concept of a Mobile User Interface

Furthermore, the user is characterized by his computer literacy. But since the PEM is deployed in all kinds of industry sectors and various occupational fields users’ knowledge about computers is not distinct. The users can be defined as advanced beginners, competent performers or experts.

Processes that are defined in TRAXION are predictable and repeatable procedures with pre-determined workflows\(^\text{124}\). Consequently, within every new process instance the user is assigned to task instances of the same task definition. Since the user works repeatedly with the PEM he possesses at least the knowledge to perform his task instance. Therefore, a general understanding of the system regarding the features that are relevant for the user can be expected.

The users are located in different hierarchy levels of the company’s organizational structure. Consequently, the user may be e.g. an operative, expecting simple and fast to perform task instances, or e.g. a manager who conducts analytical and graphically supported task instances.

In general it is to say that multiple different users with various system expectations and goals apply the PEM. The PEM is, therefore, virtually tailored to a broad user variety.

- **What are the Current Task Instances?**\(^\text{125}\)

The examination of users’ task instances results in similar findings like the examination of PEM’s users. It is characterized by a complexity of diverse task instances, whereby the Input Transforming Resources (ITRAs) are humans or machines. The task instances that are performable through PEM’s current UI are the task instances that are assigned to humans. The diversity of task instances is explainable through the consideration of the PEM’s two main task instance classifications: the group of the *PEM specific system tasks* and the group of *process’s task instances*.

The system task instances are deduced from the PEM’s system user types, as well as its system features\(^\text{126}\). Thus, three types of system task instances exist: task handling task instances, process management and control task instances, as well as the system administrative task instances. They imply different content and user responsibility. Furthermore, they are characterized by specific requirements of the PEM’s user interface.

Processs task instances are defined by multiple characteristics. One characteristic is their

\(^{124}\) Chapter 3.1.2

\(^{125}\) Hackos et al. 1998, p.23-49

\(^{126}\) Chapter 3.1.2; Chapter 3.2.1
4 Concept of a Mobile User Interface

defined duration time. Some task instances are time-critical and e.g. need to be completed within minutes. Others are non-time-critical e.g. they may take longer for the completion. Moreover, the task instance’s frequency of incidence may differ. For example, the task instance may be occurring multiple times a day or only a few times per year. In addition, they are specified through their generated and transformed data. The data differs in type, content and volume whereby the data is provided directly in specific structured UI forms or unstructured documents. The task instance, furthermore, may differentiate in its meaning and significance for the entire process instance. A task instance is assigned to only one person or a community of users. However, exactly one single person completes the task instance.

- What does the Current Task Instance’s Environment look like? 127

A task instance is never performed in complete isolation but always in a specific environment and context. Therefore, the system’s usability depends not exclusively on the task instances but also on these other two factors. PEM’s environment is defined and restricted by its need of Internet accessibility for the performance of business processes 128. Therefore, PEM is only applicable in typical office environments characterized by permanent Internet access and fixed or stationary workstations.

PEM runs on not portable PCs as well as on mobile laptops needing desks or tables as support. When using a PC, the employee is able to choose from several fixed PC locations such as office or home office. Although, he can change the location, he is still constrained to the provided work locations. Using a laptop somewhat lifts this restriction. The user is no longer forced to choose from several workstations, he himself determines the location where he wishes to work. But still, the user has to meet the requirements of accessibility to the Internet and a desk-like support. He must, therefore, always take these constraints into consideration when looking for a place to work. The deployment of a tablet PC eliminates the need of a desk but not the limitation of permanent Internet access.

4.1.1.2 M-Business as Future Factor of Influence

From the previous chapters, it is obvious that the PEM with its web browser UI already endues an adequate user interface. It covers a wide range of users and task instances in typical office environments. Therefore, at first sight, the introduction of an additional UI to further support the current users does not necessarily appear to be expedient, or particularly

127 Hackos et al. 1998, p.91-119
128 Chapter 3.2.2.1
contribute to the increase in the PEM’s usability. But, if we expand our perspective and not only examine the existing users, task instances and their environments it becomes clear that potentially other users also exist\textsuperscript{129}. These users perform task instances in a completely different environment. It is the environment of the mobile workforce.

\begin{itemize}
  \item \textit{Potential Future Users}
  
  The mobile workforce can be described as follows. On the one hand, they are employees working in customer facing processes\textsuperscript{130}. The major characteristic of these processes is the employee being the mobile factor in the customer-employee relationship. Thus, the employee must move to the customer and not vice versa. On the other hand, this mobile workforce represents employees who are constantly on the move, either inside or outside a company’s premises. In both cases the employee needs to interact with his company and he has a need to access corporate data regardless of where he is. Consequently, to achieve efficient business processes with this mobile workforce an anytime, anywhere, anyhow, communication with the company is essential\textsuperscript{131}.

  The classification of these mobile employees into job categories results in four major categories: sales personnel, service employees, consultants and traveling professionals. \textit{Sales personnel} are the employees that provide customers with product information, as well as negotiate terms and conditions. In the best case, the sales person completes the sales process by distributing the product to the customer. \textit{Service employees} frequently visit the customer performing specific operational tasks or collecting information. People who belong to this job category include field support engineers, utility meter readers, insurance claim adjusters and workers who must conduct site surveys. \textit{Consultants} provide their expertise and knowledge to the customer, and therefore support the customer in some manner. The final job category is the \textit{Traveling Professional}. Although, the aspect of traveling is also included in the afore mentioned job categories, this category primarily targets managers that mainly travel and also other people who work while being e.g. at the airport or in the train\textsuperscript{132}.

  \item \textit{Potential Future Tasks Instances}

  As seen above, the mobile workforce consists of different job categories involving a vast quantity of job specific tasks. However, it is possible to identify some major tasks.
\end{itemize}

\begin{footnotes}
\footnotetext[129]{Hackos et al.1998, p.7}
\footnotetext[130]{Turowski et al.2004, p.189-191;}
\footnotetext[131]{For further explanations see Brans 2002, chapter 3}
\footnotetext[132]{Brans 2002, chapter 3}
\end{footnotes}
4 Concept of a Mobile User Interface

Operational task instances such as the gathering of information or the maintenance of existing customer relationships is one essential task group. The initial entry of customer data is only one example for this group. Moreover, the information provision to customers or partners also exists as a task instance. Examples include the presentation of the product catalog, but also the offering of available-to-promise information. Furthermore, management task instances exist such as time scheduling, communication, as well as general management activities.

In conclusion, it is safe to say, that the aforementioned tasks, all have in common the fact that they are time-critical. Moreover, their performance by mobile employees is outside the typical office environment and inside a new environment that is the new “Mobile Environment”.

For the support of the described mobile workforce, task instances as well as the mobile environment, a new development in information technology has occurred in recent years. Business processes in combination with E-Business using wireless communications have formed the Mobile Business. The Mobile Business, also called M-Business, is the evolution of E-Business. It is used to address new customer channels and integration challenges. It is seen as the ‘next big thing’ providing essential benefits to business enterprises. These benefits derive from the enhancement of business processes through mobility and flexibility. It enables M-Business to exceed E-Business agility in a new dimension increasing business value. The new business agility may improve businesses through entry into new markets or the provision of new products and services. In addition to this, it increases revenues by reducing costs and increasing productivity. Reduced cycle times benefits customer satisfaction and loyalty. Increased customer acquisition and retention, as well as increased employee satisfaction can also be achieved.

4.1.1.3 Need for a Mobile User Interface

As can be seen above, M-Business provides immense business benefits to companies.

---

133 Brans 2002, chapter 2.6
134 Evans 2001, chapter 1.1
135 Evans 2001, chapter 1.1
136 Evans 2001, chapter 1.2
137 Hayes 2002, chapter 2.2
Consequently, by extending the PEM to support not only E-Business but also M-Business, the system’s attractiveness will be increased. The new mobility-factor will allow PEM’s deployment in processes and tasks where before, it was less efficient or even not possible at all. PEM’s mobility and flexibility can be assured by the integration of a mobile UI that allows employees to perform their tasks while on the move.

To get an idea how and to what extent this mobile UI improves PEM’s business processes a concrete example is provided for demonstration purposes. This example is taken from the job category service employees and is a prime example for the deployment of mobile UI’s in business processes. For further explanations see Turowski et al. 2004, p.196-198 and Khodawandi et al. 2004. It illustrates a process definition section involving the entry and inspection of a damage event at an insurance company. The first figure shows the original paper-based and IT independent process definition. The second figure shows the process definition managed by the PEM with a web browser UI. Finally, the last figure demonstrates the process definition managed by the PEM with the web browser UI and an additional mobile UI. For the examination, the main focus is laid on the damage inspection activity on site performed by the insurance claim adjuster.

In the first original process definition, the damage inspection activity is represented by exactly one task definition. The adjuster inspects the damage and enters the required information on paper inspection forms.

When automating the process definition using the PEM, it becomes apparent that it is not completely managed. In particular, the “examine” activity is not supported by the system.
Instead, two auxiliary task definitions have been inserted into the process definition to correct the break of information flow. Now, the adjuster always must print out the inspection forms. After the data collection, he must enter the data into the system. In general one can say that the deployment of the PEM probably results in an overall process improvement but, the specific examined process definition section - the activity inspection on site – has been degraded. Not only do the previous deficiencies still exist, but the possibility for new problems have been introduced. The lack of management of data entry increases the risk of incorrect or even omitted data. The additional new task definitions cause extra work to the employees lowering their productivity and also increasing the cycle time of this activity. Furthermore, the data transformation from electronic to paper format causes more risks. The distributed information storage results in increased data redundancy. The data transfer can cause transmission errors. Finally, it must be mentioned that the often-desired complete elimination of paper documents through process definition automation has not been achieved.

However, in the third figure, the entire process definition is managed by the PEM using a mobile UI for the analyzed activity. It is clearly noticeable that this process definition has been assimilated to the sequential course of the original paper based process. The consistent flow of information is re-established, whereby now the information is not transmitted on paper but in electronic data format. Consequently, the advent of a mobile UI results in increased employee productivity since the user makes use of the entire advantage an automated process definition implies. Moreover, the cycle time of the entire process definition decreases enabling faster response times to business changes. In addition, the adjuster cuts down his number of trips to the office. He no longer needs to pick up documents. Instead, he receives new inspection instructions on the fly. The electronic forms are updated immediately with the data on the server, leading to a faster order transaction. This again will improve customer satisfaction and therefore increase the company’s competitive advantage.

4.1.2 Interpret Requirements

In the following paragraphs, the need for a mobile UI is further analyzed to define the requirements essential for its realization\textsuperscript{138}. For this, the current and future factors of influence are taken into consideration. Furthermore, the introduction of a mobile UI represents a new communication channel between user and the PEM. Therefore, its core

\textsuperscript{138} Chapter 4.1.2
4 Concept of a Mobile User Interface

system features and underlying theoretical foundations remain the same and won’t be extended by any innovative features.

4.1.2.1 User Requirements

The best way to mobilize an enterprise is to provide mobile solutions to its mobile workforce. Consequently, CommerceQuest will gain the most success by making PEM available and usable for the defined mobile job categories: sales personnel, service employees, consultants and traveling professionals. It then can be used to manage employees in their performance of business processes and tasks in mobile environments.

In transferring these mobile employees into PEM specific system users, the following assignments are made. The mobile job categories associate mainly with the editor, reader or supervisor of the system. On the one hand, they are assigned to the system user editor since these mobile job categories very often possess the operational character in the performance of a task. On the other hand, manageable characteristics are detectable. Finally, for pure informative purposes, the system user reader should be enabled. But, it is not realistic to provide a system user administrator to the mobile workforce. The reason for this is that the system user administrator performs tasks that require the large screen of a PC e.g. the graphical design of process definitions.

As the realization of a mobile UI for the BPM is an extensive project, it is wise to first choose the most promising system users. This user is the editor since he plays the biggest role in the performance of a mobile process. Furthermore, the system user reader can easily be added to mobile UI’s future users since his features are already included in the editor’s features.

4.1.2.2 Context of Use

- Physical Requirements

A mobile application is always deployed in a multitude of physical environments. Especially in PEM’s case, as BPM software that manages a wide range of diverse business processes, one can expect a complexity of physical conditions. Considering some representatives of the mobile environment delivers a good insight. Some examples of mobile environments are the airport, restaurant, construction site, laboratory, car, hotel, park, train or warehouse. However, the mobile use of the PEM can also be envisioned in the traditional environments such as the office, conference room, or home office. Thus, the mobile UI must deal with a large number of physical constraints. First of all, it must be
usable regardless of the available space. Then, since some mobile tasks are performed while the user is standing, the existence of a desk cannot always be guaranteed. Sometimes, the user is exposed to heavy background noises, and then again the user must work without any light. Moreover, the work environment’s temperature can be extremely low or extremely high. Furthermore, it is worth mentioning that the user may work in dirty environments putting the deployed device into danger.

- **Social Requirements**

These requirements define the aspect of collaboration and coordination with the system. Since PEM is software managing collaborative business processes, this issue is, for the most part, already ensured by its back-end system. But with the advent of a mobile UI some new social issues arise. On the one hand, the mobile UI will be used to work with partners or customers, showing them important information on the device. On the other hand, the added mobility aspect of the PEM results in an altered manner of collaboration. In future, the collaboration will no longer be only synchronous, but also asynchronous.

- **Technical Requirements**

The main physical constraints derive from a mobile device’s characteristics. Contrary to the traditional PC, a mobile device possesses only a small screen, little available memory and limited processor power. Furthermore, these devices differ in their operating systems. Besides this, mobile devices cannot necessarily rely on permanent Internet accessibility. Whereas, private networks assure uninterrupted Internet access, networks by public providers currently do not provide fully developed wireless network coverage. Finally, it should be mentioned that these devices must be charged frequently where power sources in the mobile environment are not always available\textsuperscript{139}.

### 4.1.2.3 Data Requirements

The introduction of a mobile UI does not cause a change in the PEM’s fundamentals. Consequently, PEM’s business process and task definitions, as well as process management rules remain the same. This leads to the conclusion that also task data types do not change. The new mobile UI will handle the same task data as the web browser UI. However, one restriction of PEM’s data is its screen presentation form. Because of current mobile device limitations, within the mobile UI it is only possible to show data in specific user interface forms and not as documents. Finally one must consider the fact that the data

\textsuperscript{139} Mallick 2003, p.14-20

52
4 Concept of a Mobile User Interface

consists of sensitive, corporate data. The unauthorized access of this data must be forbidden.

4.1.2.4 Functional Requirements

In the mobile UI’s design, the role of the future mobile user, the task editor, is the decisive factor. Consequently, the mobile UI’s functional requirements must be tailored to this user with strong focus on his tasks, their context and the data that is processed.

As an all-purpose user interface, the mobile UI must be made available in online mode, as well as in offline mode. This feature enables a PEM deployment regardless of Internet accessibility. On the one hand, when the user has permanent Internet access, he must be able to work remotely in Real-time on the tasks of PEM’s back-end database. On the other hand, when the Internet accessibility is only limited or very expensive, On-Demand-Online access must be provided to the user. The user goes online to download his tasks. He works offline on the tasks stored on the mobile device and later synchronizes these tasks with the PEM engine. With this feature, the mobile UI will remove the previous constraints of stationary locations and therefore will integrate perfectly into the users’ current environments.

Contrary to the PC with its possible multi-user deployment, the mobile device is created for a single user. Consequently, PEM’s mobile UI can be personalized with regard to its data and functionality. By defining an adequate selection of PEM’s interaction operation features described in 3.2.1, the system’s complexity is managed and the focus is exclusively laid on the essential functions. For this, the mobile UI must offer PEM’s traditional features that match the system users editor and reader. These features are Authorization and Authentication, as well as Task Handling. Authorization and authentication must be realized as fundamental conditions since a secure connection to the back-end must be assured. Furthermore, task handling must enable the user to claim, edit, save, complete, release and undo tasks. Finally, deriving from PEM’s offline capability, features to download and synchronize tasks must be added. As PEM’s primarily usage is still the management of structured, predefined processes, no other features such as e.g. email or calendar ability will enhance this mobile application.

4.1.2.5 Usability Requirements

Software development should always strive for a usable interactive product. Therefore, the requirements for usability - the usability goals - must be defined and their relevance must
4 Concept of a Mobile User Interface

be proven for the new mobile UI\textsuperscript{140}.

- **Learnability**

Since PEM’s future users are on the move, they are mainly far away from companies’ headquarters or they only visit them infrequently. Therefore, the user is unable to rely on human support when learning the system. Instead, he learns the system by exploring it or by studying help documentation. The mobile environment is further characterized by high speed and limited time. The available time must be used productively. Taking this into consideration, it is essential to spend as little time as possible on secondary activities such as learning a system. Consequently, understanding the mobile UI must not take too much time. Instead it should be fast to learn and easy to remember.

- **Efficiency**

After having learnt how to use the mobile UI, it should provide a fast and efficient way to perform tasks. The efficiency goal plays a major role within a mobile environment. It helps to save time and reduce cycle times within mobile processes, and therefore increases productivity and reduces costs. Achieving great efficiency in core processes amplifies these benefits even more. The efficiency of mobile applications is increased by short start-up and shut down times, standby mechanisms, as well as the efficiency of the general application usage. One example of this is the deployment of PEM’s mobile UI in hospitals. Here, the nurse uses the PEM to gather patient data. While on the move from one patient to the next patient she must enter data in a very short time. An inefficient application would not survive in this environment.

- **Effectiveness**

The mobile environments’ characteristic of unstable Internet access and the multitude of mobile devices makes effectiveness a very essential goal. Whereas PEM’s web browser UI is permanently executable because of wired network capability, the mobile UI cannot rely on this. Therefore, if the mobile UI does not assure an offline usage, PEM won’t be able to adequately support the mobile user in its typical environment. For example, processes will halt until Internet is accessible again. This will lead to inefficient and unreliable processes. Beside this, the effectiveness goal must be considered for the mobile device diversity. The application may be executable on one device type but at the same time inoperative for another one. However, the effectiveness regarding the Internet accessibility and the

\textsuperscript{140} Chapter 2.2.2.2

54
effectiveness regarding the device choice are very often in contrast.

- **Memorability**

The use of the system must always be easy to remember. In particular in customer facing processes, the employee must present a certain confidence in using the system. If the employee does not know how to use the system, he will look incompetent. This will lead to a decrease of customer’s confidence in the employee’s expertise and skills. In addition to this, a difficult to remember mobile UI will also be less efficient. Every time the user applies the mobile UI, he must relearn it.

- **Safety**

The before mentioned employees’ absence from headquarters also influences this goal of a system’s safety. It can be explained as follows. In a mobile environment the user strongly depends on fast external help. But, he can rarely rely on it. Therefore, the mobile UI should be extremely reliable and prevent the user making errors. If an error occurs, the UI must possess the ability to help the user to recover. If safety is not ensured and an error did occur, the employee is forced to interrupt his activities, and he must return to his office looking for the system administrator’s support. The consequence of this is an interruption of the process flow that will raise additional costs.

- **User Experience**

The introduction of a new complementary UI to an already existing UI enforces the user to adjust to new working methods. If this leads to immense efforts and discomfort by the user, he will not accept the new mobile UI. Instead, he will fall back to using the previous web browser UI. Thus, the user will still conduct his old working method of entering the data on paper and afterwards reentering it into the system. Therefore, the benefits of a new mobile UI will not be reaped.

### 4.2 Conceptual Design

Mobile application development is entirely different from the application design for a desktop computer. This derives from the traditional “default” conditions the PC as standard device holds and the mobile device cannot rely on. In particular, the office as environment, the large screen size, the powerful processor, the immense memory space and permanent Internet connectivity are not available anymore. Instead, the mobile application is characterized by diversity but also restrictions in these conditions. Therefore, when designing a mobile application more than the regular design activities represent the
conceptual design process. On the one hand the available devices, and on the other hand, potential application solutions must be analyzed and proven for the defined usability goals. However, because of the mobile devices’ characteristics as well as the rather abstract application solution’s nature not all usability goals can be examined and assured in this phase. Some goals can only be achieved in the mobile UI’s real design phase that is the physical design in this concept. In addition to the examination process, the best fitting combination of both with regard to the evolving usability goal trade-offs will be chosen as end solution of this conceptual design phase.

4.2.1 Device Classification

In recent years, an immense development in mobile technology has occurred. The mobile landscape is changing rapidly and the variety of mobile devices is becoming increasingly complex. The only feature these devices all have in common is their portability\textsuperscript{141}, whereby the degree of portability differs. In general, coherence between the device’s available features and its portability is discernible.

![Figure 15 – Mobile Device Classification](image)

Nevertheless, mobile devices can be categorized into two major groups: on the one hand, laptops and tablet PCs, and on the other hand handhelds. In short, the first category possesses similar features as a standard PC except that they are mobile. The devices in the second category only offer a carefully selected subset of desktop computer’s features. It leverages them to become extremely mobile and interesting as skeletal structure for PEM’s mobile UI. Their mobility is characterized by the operation with only temporary cables,

\textsuperscript{141} Mallick 2004, p.4
easy usage without rests on a table and the ability to connect to the Internet with or without additional applications\(^\text{142}\). Comparing these characteristics with the previously defined requirements, it becomes obvious that handhelds’ characteristics and the requirements match. Furthermore, since PEM’s web browser UI already runs on laptop and tablet PC when a stable Internet connection is assured, an additional mobile UI that only runs on these devices would not result in a large usability increase. Consequently, a pre-selection of the before presented mobile devices already eliminates laptop and tablet PC as potential future devices. In the following, the different handheld types are further analyzed with the goal to find the best fitting device for PEM’s mobile UI solution.

Handhelds are categorized into Cell Phones, Pagers, Personal Digital Assistants (PDAs) and Communicators. The first three are the original handheld types. In the beginning of mobile device development, they held functions that were exclusively specific for these types. Nowadays, an increasing number of communicators arise. Communicators represent a hybrid type offering the functionality of all three types\(^\text{143}\).

**4.2.1.1 Cell Phone**

Although the cell phone only provides limited features, it is the most portable among the devices. The data input occurs by using the user’s voice, the numeric keypad, or a keyboard as an add-on. The data output is delivered via voice, or a very small display. The original cell phone only has a very limited processor and low memory space. Internet connection is achieved through wireless networks. Its primary use for a long time has been the pure voice communication between two people. With improving technology, the cell phone is nowadays used for much more. They still offer the user voice calls, but also caller ID, voice mail, short messaging service (SMS), basic calendar functions, address lists and small cameras\(^\text{144}\). A further capability is the cell phone’s Wireless Application Protocol (WAP) browser that allows a limited form of Internet browsing\(^\text{145}\).

---

\(^{142}\) Weiss 2002, p.4-20  
\(^{143}\) Classifications of handhelds according to Weiss 2002  
\(^{145}\) Lehner 2003, p.142-147
4 Concept of a Mobile User Interface

General Usability Goals:

**Learnability:** The first time when using a cell phone, the user must familiarize with its menu. One difficulty for the cell phone’s learnability is its small screen size that causes a very high risk to get lost in the application. Furthermore, the different available cell phones don’t provide a common menu. Therefore, the change to a new cell phone does not allow reusing the knowledge gained by using a previous cell phone. However, the learning curve of an application that is deployed on the cell phone depends entirely on the application itself.

**Efficiency:** The deployment of cell phones as voice interface is definitely efficient when it is used for a human-human communication. In a human-machine communication, the efficiency entirely depends on the defined course of communication. In case of a data interface, the efficiency goal is a critical factor. On the one hand, the cell phone’s small screen limits the visible data complicating the navigation through the application. On the other hand, the input with the numeric keypad is cumbersome and even add-ons cannot counteract this laborious data input method. Consequently, cell phones are not suitable for the entry of high volume textual data.

**Effectiveness:** A cell phone is effective when a wireless network is available, otherwise only the features that don’t require network connectivity are enabled to run on the cell phone. Furthermore, its long battery life contributes to continuous use.

**Memorability & User Experience:** These goals completely depend on the specific application and cannot be generally defined for this mobile device type.

4.2.1.2 Pager

Compared to the cell phone, the pager possesses more features. But at the same time, it is also less mobile because of its bigger size, larger screen and more complex keyboard. Instead of a numeric keypad, a pager provides QWERTY keyboard. This keyboard is similar to the desktop keyboard, although it is utilized only with two thumbs. In the USA pagers offer an always-on wireless network connection for one fixed price. A pager allows two-way messaging. Emails can be sent and also directly received from the server. Receiving emails, does not take any actions by the user since they are automatically pushed on the device. In current developments, pagers are increasingly equipped with additional features such as contact management and calendar.\(^{146}\)

---

\(^{146}\) Beaulieu 2001, chapter 1.3; Hayes 2002, chapter 10.3.1
4 Concept of a Mobile User Interface

General Usability Goals:

Learnability: In this aspect pagers are similar to cell phones. It will take the user a while to become experienced with this device. A change to a new pager requires a new menu to be learnt but the input methods remain the same.

Efficiency: When experienced in two-thumb typing, the user is able to entry a high amount of data in a very short time. In comparison to the cell phone, the bigger screen supports the ability of a faster navigation through the application. Moreover, being always online, the user is always provided with the newest data. This helps him to be able to react fast to changes such as the rescheduling of appointments.

Effectiveness: Similar to the cell phone, a pager is effective when Internet accessibility is guaranteed otherwise only the network independent features work.

Memorability & User Experience: These goals are increased through a rather graphical and visual data presentation, as well as user friendlier text input mechanisms. Other factors derive from the design of the prevailing application in combination with hardware characteristics.

4.2.1.3 PDA

The Personal Digital Assistants (PDA) belongs to the group of advanced handhelds. Characteristic for this device type is its large screen with touch sensitive capability. The user interacts with it by using a stylus pen or an additional keypad. Furthermore, it has the most complex OS among the handheld types. Because of its relatively high computing power and large memory space, it has become very popular for the mobile workforce. The basic functions of a PDA are Calendar, Directory, Notes, Calculator, Alarm and To-Do lists, as well as desktop synchronization147.

General Usability Goals:

Learnability, Memorability and User Experience: In general, the PDA’s large touchscreen and the stylus pen make the device easy to work with, and allow a fairly sophisticated User Interface. It achieves all three usability goals.

Efficiency: Providing the user with the ability to choose between different input mechanisms allows a constant adaptation to existing requirements. Thus, the user can always apply the most efficient usage with regard to the respective conditions. Furthermore, the device’s high processing power will enhance working with the PDA.

---

147 Hayes 2002, chapter 10.3.2; Weiss 2002, p.33-36
Finally, PDAs are rather cost-inefficient because of their high cost.

**Effectiveness:** On the one hand, the mobile storage capability enables a deployment regardless of available Internet accessibility and therefore positively influences this goal. Moreover, the selection of different wired or wireless Internet connection mechanisms not only makes this device more efficient but also more effective. However, on the other hand, the energy intensive touchscreen and processing power give the device a rather short battery life.

### 4.2.1.4 Communicator

The communicator is also defined as an advanced handheld. It presents the most multifaceted device among the handhelds since it combines cell phones, pagers and PDA capabilities in one single device. For this, it either holds the physical appearance of a pager, or a PDA with the integration of a cell phone. A communicator may possess the voice ability of a cell phone, the two-way-message mechanism of a pager, and at the same time the OS, processing power and memory space, as well as the ability to install third-Party application of the PDA. In some cases, all capabilities are realized, in others, some capabilities are neglected\(^{148}\).

**General Usability Goals:**

In general, it is to say that Communicators mainly take over all positive factors influencing the different usability goals of the other device types. However, because of their physical conformity with pagers and PDAs, they keep the short battery life of the PDA that counteracts the device’s *Effectiveness*. Moreover, when used as cell phone it may be rather cumbersome and difficult to use because of its bulk size. This may cause a decrease in the device’s *User Experience*. The Communicators are also somewhat expensive.

### 4.2.2 Application Solutions

For the design of PEM’s mobile UI, two extreme computing paradigms exist. These are on the one hand *Thin Clients*, and on the other hand, *Smart Clients*\(^{149}\). Thin clients are further categorized into thin voice clients and thin data clients. A thin client is characterized by no or little client deployment, as well as few capabilities and simplicity. A smart client is more complex and requires client software but also provides more extensive capabilities.

---

\(^{148}\) Weiss 2002, p.36-38

\(^{149}\) Classification according to Mallick 2003, p.171-376
4.2.2.1 Thin Voice Client

For the user-application communication voice clients provide a voice interface instead of a visual interface. Using a phone that is stationary or mobile, wired or wireless, the user interacts with the system. The course of communication is mainly menu-driven, whereby the user talks to the system selecting words from multiple options. For this, the user follows these options in the predefined sequence the application provides. However, the voice recognition technology that is applied for this communication is still in its infancy. Very often the dialog must be limited because of the not immature technology. But one can expect that with increasing development of these technologies more complex and intuitive dialogs can be designed. Since the user is enabled to connect to the application through every device that enables voice capabilities, true universal access is given to the application. The architecture that underlies this interface is very similar to that of Internet applications with web browsers as UI such as the Process Execution Manager. In contrast to these Internet applications, not a web browser but a voice browser using the voice as data format is provided\(^{151}\).

General Examination of Usability

*Learnability:* A voice interface has a very short learning curve. This derives from the fact that the user does not need to know how the system works. He only must follow the instructions. Therefore, no special training is required.

*Efficiency:* For the same reasons a voice application is easy to learn, it can be also less efficient. Especially when the user wishes to go fast through the dialog. He must always follow the pre-determined sequence. Within each option the user must listen to the entire instruction before he can continue. Providing good shortcuts that accelerate the navigation

\(^{150}\) Mallick 2003, p.93

\(^{151}\) Mallick 2003, p.361-376
may counteract this behavior.

**Effectiveness**: Being accessible everywhere and independent of the phone device makes this UI in many cases very effective. For example, while being on the move it is useful to apply voice instead of hands. Furthermore, possessing the ability to apply any kind of phones makes this UI a cost-efficient UI. On the other side, it is not eligible when an interface is needed that must allow the entry of unstructured voice for data collection purposes. Then the UI must not only recognize word options, but also must convert voice into text. As mentioned before, for this purpose, the current voice technologies have not been completely developed. Finally, a thin voice client is not usable in loud and noisy environments.

**Memorability**: A sophisticated application navigation with easy to recognize options facilitates the UI’s usage. However, a not well-implemented UI with inadequate and indistinct option commands forces the user to remember his way through the application. In this case, in particular when the UI is infrequently used, the user must relearn the system every time he applies it.

**User Experience**: This goal also depends on the application’s sophistication. But additionally, the user’s characteristics and preferences play an essential role. For example, users who possess visual rather than aural abilities will rather prefer a thin data client.

### 4.2.2.2 Thin Data Client

This client also resembles a desktop’s web-browser interface, not only in its behavior, but also in its underlying architecture. Here, a micro-browser is applied for the presentation and entering of data. The client’s functionality, its business logic and content generation routines are completely located on the server. Server and client communicate with each other through different markup languages over HTTP\(^{152}\).

**General Examination of Usability**

**Learnability**: Since the traditional PC user is accustomed to the web browser, the change to the micro-browser is rather facile. Therefore, the learning curve is fractional.

**Efficiency**: This usability goal depends, on the one hand, on the behavior of the respective application, and on other hand, on the screen size and input mechanism of the applied device. Since a micro browser is device-independent the user is able to choose the device that best fits his needs. Furthermore, network coverage, penetration and especially

\(^{152}\) Beaulieu 2001, chapter 5.4; Mallick 2003, p.281-360
bandwidth influence this client’s efficiency. However, when online the user can make use of real time access. Fast data processing is therefore guaranteed.

**Effectiveness:** There are several reasons for a less or more effective thin client. First is the availability of Internet access, as without Internet accessibility, the application does not run at all. However, since it doesn’t require much processing power or memory space, there are fewer resource restrictions on the required devices, making this UI usable on a broad range of devices.

**Memorability:** An easy to learn system also implies that it is easy to remember because there are only a few things for the user to remember.

**User Experience:** This goal strongly depends on the chosen device and the nature of the deployed application. An unintelligible application will cause the user to get lost in the application and will decrease his satisfaction. In addition, the user experience will also fall where the user always struggles to find network connection and is unable to use the application.

### 4.2.2.3 Smart Client

A smart client is a powerful alternative to the thin clients for voice and data. It is a customized standalone application running on PDAs or communicators. A smart client functions as follows: The data is downloaded and stored on the mobile client and then processed while disconnected from the server. When Internet connectivity is available the data on the server is updated through synchronization. By offloading business logic and application data onto the client, there is a high independence from the back-end system, and therefore, removes some of the restrictions of a thin client application. When developing a smart client, it has to be taken into consideration, that one can expect extensive adjustments in the back-end system, as well as a complex and innovative client development\(^{153}\).

**General Examination of Usability**

Resulting from the previous characteristics, the designer gains a higher discretion in the application’s development. Most of a smart client’s usability goals such as, in particular, **Learnability** and **Safety** are strongly influenced by the designer’s design freedom, and therefore these goals differ between the prevailing smart client applications. Beside this, there are also other factors that define the usability of a smart client.
4 Concept of a Mobile User Interface

Efficiency: Since a smart client is not restricted to a purely online usage, it can also be applied when it is disconnected from the server. In that case, the client’s promptness is not restricted by the network bandwidth, and therefore works faster. Furthermore, this characteristic also implies a more cost efficient usage. The fact that a smart client requires devices with adequate processing power, storage availability, as well as an Operating System and may be Java enabled, allows for the development of more user friendly UIs.

Effectiveness: The on-demand Internet access feature of a smart client, furthermore, involves a complete autonomy from environmental restrictions and enables the user to deploy this application anywhere he wishes, wired or wirelessly. However, he is restricted to a certain range of devices since it cannot be implemented on low-level handhelds.

Memorability: Because of a smart client’s extremely high design freedom, the application can conform to its desktop’s standard UI. Therefore, by applying the consistency heuristic, the user can reuse his knowledge from the desktop UI and will easier remember the UI. However, a bad design also leads to a difficult to use application.

User Experience: This goal is influenced positively by the few restrictions on the smart client’s. In general, the deployment of high-level devices, the on-demand Internet accessibility and the freedom in design improves the user’s feel of the application.

4.2.3 Future mobile User Interface

Superior objective of the introduction of a new mobile UI is PEM’s enlargement to a broader user range achieved by an increase of system acceptability. The intended users are mobile employees that will use the PEM under completely new and diverse conditions. Therefore, the achievement of best usability is essential. One solution must be found that covers most mobile application fields and constantly adapts to the prevailing user, situation, environment and resulting conditions. However, when combining device types and application solutions it becomes obvious that one single ideal solution does not exist. Usability trade offs arise. One example is the thin voice client execution on an analog phone for the entry of high data volume. This UI is effective since it can be used everywhere, but it is also inefficient since a voice interface is not eligible as data collection UI. Another example is a thin data client that runs on nearly any device. It improves user experience, as the user feels free to choose the device that best fits his requirements. But, it becomes ineffective as soon as the Internet connectivity is not available anymore.

The described examples clearly show that simply striving for the fulfillment of all usability goals does not automatically lead to one solution. Therefore, the importance and relevance
The effectiveness of the mobile UI implies two contrary restrictions depending on the application solution. This is on the one hand, the thin data client’s limitation of needing permanent wireless network connectivity, and on the other hand, smart clients’ need of advanced handheld types comprising PDAs and communicators. Whereas the network coverage must be taken as given and cannot be changed, the device can be chosen by the user. Consequently, a mobile UI’s realization as a smart client is more effective since it is with the adequate device – constantly usable regardless of network coverage. In addition, effectiveness is further extended by the smart client’s implied high-level handhelds. One factor for this effectiveness rise is the advanced handhelds’ multiple wired, as well as wireless, connection mechanisms. First of all, wireless Internet connectivity can be achieved through an integrated wireless modem. Secondly, the PDA accesses the Internet through other wireless network-enabled devices e.g. a cell phone. Moreover, the device can achieve Internet accessibility through a wireless LAN (WLAN) card that connects to a wireless network. However, wired connectivity is established through a cradle to a regular modem, or directly to a PC. Furthermore, the short battery life of the PDA and the communicator counteracts the effectiveness goal. However, the provision of spare batteries eliminates this concern. In contrary to the effective smart client – advanced handheld combination, the thin voice client – phone/cell phone combination also exist. As already said earlier, this voice client is very effective to be applied anywhere. But since not only information-less but also high data volume tasks are managed by the PEM, this UI is not adequate.

A further examination of the smart client – PDA/communicator combination with the consideration of the remaining usability goals leads to following results. The efficiency is highly improved by the high-level handhelds various input mechanisms such as the
touchscreen with the pen. In particular, the combination of both, input and output mechanisms, positively affects the learnability, user experience, memorability of the mobile user interface. In general, the considerations to the fulfillment of these goals directly derive from the before analyzed usability goals for the PDA and communicator, as well as the smart client.

### 4.2.3.2 Evolution to a Thin-Smart Hybrid

The realization of a typical smart client in combination with an advanced handheld shows altogether good effects on the mobile UI, but it also causes one disadvantage for PEM’s deployment. The ability to store data with subsequent server synchronization makes this solution very efficient for the on-demand online usage. Nevertheless, for the real-time data access it is rather inefficient. Here, a thin client is definitely more eligible. One example is the unnecessary data storage on the device. Limited memory space is occupied needlessly and more data needs to be transmitted than with a typical thin client. Therefore, to provide an efficient solution regardless of on-demand or real time access, the solution must be realized as a hybrid of a thin and a smart client. For this, the smart client has to be extended by thin client’s basic principle of real-time access. Consequently, the hybrid will be enabled to behave like a smart client, as well as a thin client. It constantly adjusts its client behaviour to the user’s current needs and situations. However, to ensure a thin-smart hybrid the mobile UI must possess the underlying component architecture of database and business logic on the client side. Furthermore it still imposes the implicit hardware requirements of a smart client.

The described thin-smart hybrid executable on high-level handhelds approaches the defined usability goals very well. Moreover, it provides good chances to further converge these goals in the following physical design phase.

### 4.3 Physical Design

The previous phase particularly focused on the assurance of the effectiveness goal. Furthermore it provides a good basis for the usability goals efficiency and user experience. However, with mobile UI’s physical design the fulfillment of these usability goals, as well as the earlier slightly neglected goals of learnability, memorability and safety will be further enhanced. Striving for a user-centered interface, the physical design is presented as a manifest model. Consequently, the main focus is laid on the close approach to user’s
4 Concept of a Mobile User Interface

mental model\textsuperscript{154} emphasizing his future understanding of PEM’s mobile UI. The rather technical UI descriptions are described in the following chapter to the prototype implementation.

4.3.1 Fundamentals

In the following, the mobile UI’s extended fundamentals comprising its change of system states and the new system user are introduced.

4.3.1.1 Online & Offline Mode

The major characteristic of the thin-smart hybrid is the capability to work efficiently connected to, as well as disconnected from, the server. For this, the UI must possess the ability to be set in different system states, referred to as modes. “A mode is a state the program can enter where the effects of a user’s actions changes from the norm – essential a behavioral detour”\textsuperscript{155}. Since PEM’s mobile UI behavior alters when the user connects to the server, two modes exist: \textit{Offline Mode} and \textit{Online Mode}\textsuperscript{156}.

The offline mode is defined as default mode, as it is UI’s initial mode. This mode is applied for the on-demand-online access when no permanent Internet connection is available or it is very expensive. Applying this deployment type, the user works on his pre-downloaded tasks disconnected from the server. The available data and functions in this mode are limited because the user is restricted to the access of data that is stored on his device. Thus, the functions he performs in offline mode only affect the data on the mobile device. An anomaly of this mode is that it is not functionally entirely isolated. At a particular time in the task performing activity, the user must be in online mode. This is at the beginning when the user pulls task data on the client, and at the end when he sends the completed tasks back to the server.

However, the online mode does not necessarily require the offline mode since it can be applied in two different ways. Beside the described on-demand-online procedure, it is possible to use the mobile UI for real-time purposes. While the user is in online mode, he accesses remotely the back-end data. Thus, tasks are performed and completed directly on the server.

To provide the user with a high independence of mobile UI’s deployment, the application does not automatically change from one mode into the other. The required action for

\textsuperscript{154} Chapter 2.2.1.3
\textsuperscript{155} Cooper 1995, page 69
\textsuperscript{156} Chapter 4.1.2.4
4 Concept of a Mobile User Interface

Changing modes\textsuperscript{157} must be performed by the user. In case a network connection is assured, the user initiates a login procedure to PEM’s back-end. A successful authorization and authentication sets the mobile UI into online mode. The UI is reset into offline mode either by the user himself, or by the system when network connectivity is not available anymore. In online mode the secure access of server’s data is provided directly by the mobile UI. However, the security of task data stored on the mobile device is controlled by the standard security mechanisms implied by the mobile device.

The utilization of two different modes in PEM’s mobile UI is indispensable to meet the requirement of multiple deployment options. But, it also involves a risk in assurance of the mobile UI’s usability. This derives from the fact, that modes are always sources of error since they confuse the user with their implied change of system behavior. Therefore, for the design of PEM’s mobile UI, it is of great importance to fulfill the usability goal safety. For this, the heuristics feedback, preventing errors and error messages are applied in the further mobile UI design.

4.3.1.2 Mobile Editor

Referring to the new user requirements defined in chapter 4.1.2.1, PEM’s mobile UI’s future user is mainly the system user editor. But since this user now operates in a complete different environment, his tasks and conducted functions must be adjusted to this new context. Therefore, it is not possible to simply take over this system user and apply him for the mobile UI. Instead, a new user must be created. This new system user is the mobile editor. The mobile editor is a specialization of his superior system user editor. Differences between these two users arise from, first of all, the mobile applications’ general main principle being a personalized application; secondly, the device limitations of memory space, processor power, and screen size; and finally, the applied usability heuristic of simplicity. Consequently, the mobile UI is designed to perfectly meet the mobile editor’s specific needs saving resources and keeping the UI simple. Only the data and functions that are relevant for this system user are provided by the mobile UI. Chapter 4.3.2 further differentiates the data and functions that are available for this new system user.

4.3.2 Task Data

The task instances that are accessible to the mobile editor are only a subset of the task instances provided to the original editor. An analysis of the instances relevant for the original editor helps for further definitions.

\textsuperscript{157} Raskin 2000, p.45
4 Concept of a Mobile User Interface

The editor uses the web UI to handle his personal task instances. Therefore, he comes in contact exclusively with the instances that are either activated, or claimed by him or by other users. They are allocated to the task list My Tasks. Since the mobile editor will also apply the mobile UI for task performing purposes these task instances are highly relevant for him. Beside the access of My Tasks, the original editor accesses his or other users’ completed task instances allocated to the task list Completed Tasks. However, the provision of these instances to the mobile editor will not lead to an increase in mobile UI’s usability. This is because the mobile editor does not need to access these instances. Therefore, the provision of the completed task instances would only unnecessarily raise the UI’s complexity and counteract its simplicity. The task instances that are accessible by the mobile UI are referred to as Mobile Tasks.

This group of task instances further differs in their meaning and performable functions within the mobile UI. This mainly derives from their new two storage locations. Whereas on the one hand, the entire task instances still exist as originals on the server, on the other hand, a copy of a data-subset of some task instances can be stored on the mobile device. In the following, the task instance’s characteristics and functions depending on the two storage locations are described.

4.3.2.1 Task Lists on the Server

In online mode, the mobile editor is enabled to access the original Mobile Tasks located on the server. For this online access the mobile UI provides an extra task list that is defined as All Tasks. It is used for the real-time but also partly applied for the on-demand online access providing the user with the available task instances and executable task functions.

When applied for the real-time access, the task list is similar to the task list My Tasks of the web browser UI. It shows the same task instances and functions. Furthermore these
functions comprising claim, release, save and complete equally affect the original task instances on the server. Changes being made are immediately visible to other users, and where a task instance is completed, the respective process instance will be immediately continued.

However, using this task list All Tasks for the on-demand-online deployment, the mobile editor selects the required task instance originals and downloads copies of these instances on the mobile device. It is not possible to download multiple copies of a single task instance onto the mobile device. This prevents errors since the data on the mobile device remains free of redundancy.

4.3.2.2 Task Lists on the Client

As already said before, for the on-demand-online deployment, the user must have downloaded the task instance copies to be able to work in offline mode. When these copies are first saved, all task instance copies are in the same state: every copy is claimed by the mobile editor and have not been completed. In the further task handling, the mobile editor changes the copy’s data, saves it, and at the end he completes it. Consequently, the task instance copies on the device are either of the state uncompleted or completed. However, because of the task instance copies’ two variant states, their meaning for the user, as well as the executable task functions differ. Therefore, the allocation of these task instance copies to one single task list complicates the mobile UI’s usage. This is because in general the user either accesses his uncompleted or completed task instance copies. Thus, he is not interested in seeing all task instance copies regardless of their state. Furthermore, he does not want to search every time for the required copies of the needed task instance status. Moreover, displaying all task instance copies on an already small display will decrease the system user experience counteracting the usability heuristic of the minimization of memory load. Beside this, errors may occur if the user performs mistakenly a function not applicable for the selected task instance copy.

For these reasons, it is essential to design two local task lists: the *Current Tasks* that consists of the uncompleted task instance copies, and the *Completed Tasks* for the completed task instance copies. This allocation of task instance copies to two task lists is also consistent to the task lists My Tasks and Completed Tasks of the web browser UI. Therefore, the change from the web browser UI to the mobile UI is facilitated for the user since he probably accepts these familiar task lists very quickly. In addition a short learning process of the task lists can be expected.
Since one of the mobile UI’s requirements is the constant adaptation to the existing situation, this local task list changes its behavior depending on the application’s mode. Consequently, in offline mode the available functions are adjusted and limited to only the needed functions. With this limitation the simplicity of the UI is assured and errors are prevented. Thus, the functions save and complete are performable whereby the function complete shifts the completed task instance copy into the task list Completed Tasks.

In online mode, the mobile UI does not only access the Current Tasks. With every change of the task instance copy also the original task instance is changed and updated on the server. For example, if a task instance copy of Current Tasks is completed in online mode, it is not shifted to the local task list Completed Tasks. Instead, the changes being made are updated on the original task instance, which is then directly completed on the PEM engine. Afterwards the task instance copy is removed from the mobile client. The mobile UI behaves similarly in online mode with the functions save and release.

Within this task list, the user is enabled to perform changes on his locally stored already completed task instance copies. Here, the user undoes a task instance copy shifting it on the mobile client from the Completed Tasks to the Current Tasks. Furthermore, he is enabled to open a task instance copy to reedit and save its data. Completed Tasks’ available task function always behaves the same regardless of the existing application mode. The completed task instance copies are updated with server’s original task instances when a synchronization process is initiated.
4 Concept of a Mobile User Interface

- *Synchronization Process*

The two storage locations of a task instance – on the server and on the mobile client - may cause the following conflict: When the mobile client is in offline mode and therefore disconnected from the server, the original task instance on the server can be changed without updating its copy on the mobile client. Thus, the task instance copy is not correct anymore. Nevertheless, the mobile editor continues to work on his incorrect copy not knowing that the data has been changed. When the mobile editor connects to PEM’s engine to update the original task instance with the task instance copy, the conflict arises.

Figure 21 – Task Instance Synchronization Conflicts

For the resolution of this conflict, it is important to define and assure synchronization rules. Here, the major rule is that the original task instances changed by a supervisor on the server always have a higher priority than changes being performed by the mobile editor on the task instance copy. In case of such a conflict, the user will be informed and must manually delete the affected task instance copies on the mobile client. In case of no conflict, the original task instances on the server must be updated with the changes of the task instance copies stored on the mobile client.

As already said before, the task instances are automatically synchronized when the user performs a function of the Current Tasks task list in online mode. In addition to this automatic update mechanism, the user can instantiate manually a synchronization process. On the one hand, this process checks if the task instance copies of the Current Tasks are still consistent with their originals on the server. On the other hand, it updates the original task instances with the task instance copies of the Completed Tasks.

Nevertheless, the user is always informed about the synchronization’s success or failure. In
case of a failure the user is provided with adequate feedback and receives detailed error messages for the failure reason. This behavior increases the safety of the mobile UI. Furthermore, the user experience is raised. This is because the user is always informed about the mobile UI’s background activities. It counteracts the user’s uncertainty of the performed function’s success caused by the possibly small bandwidth implying longer response times for the communication with PEM’s back-end.

4.3.3 Screen and Navigation

In the following, mobile UI’s screens, as well as its screen navigation structure is introduced. Main focus of the screen and navigation design has been the utilization of the defined usability heuristics. Furthermore, the screens are optimized for the handhelds’ specific screen type and created to assure the mobile UI’s consistency with the web UI. Thus, first of all, these requirements are explained.

4.3.3.1 General Screen Structure

Advanced handheld’s screens completely differ from PC’s screens. The PDA’s and communicator’s possess a very small screen, whereby the largest of these screens contains as many as 160 x 160 pixels showing 20 lines of 20 characters per line. A further characteristic of this screen is its direct manipulation. For this, the touch-screen is used as input as well as output element. In addition, it must be said that the screen is not necessarily capable of displaying colors.\(^\text{158}\)

For a simple to read and usable display a standard screen structure is often applied by mobile applications. This standard divides the screen into several partitions. Once, the user understands these partitions within one application he can easily carry over this knowledge to other similar structured mobile application. The application of this standard design simplifies the learnability of PEM’s mobile UI.

The structure is defined as follows. The upper area is reserved for the screen title. However, behind this title a menu bar is hidden. It becomes visible when the user clicks on the title. Then, it shows commands such as the navigation to other screens or other functions the user can perform. The major area of the screen is presented by the middle part. It is the intrinsic working area for the user. It is either used for navigation

---

\(^{158}\) Weiss 2002, p.50
purposes, or for data collection and data presentation. Using this area for navigation, a menu element is placed into it. For data collection and presentation, a form element is used. In this form element fields comprising labels, text fields, date fields, combo boxes, and so on are located. In the lower area soft-buttons are visible. Soft-buttons are buttons that are placed directly on the touch-screen whereby they provide the only possible shortcuts of the mobile UI. This is because in mobile devices, it is not possible to use the typical keyboard shortcuts.  

4.3.3.2 Web UI Consistency

One of the defined usability heuristics is consistency. Consistency can exist between different UIs but also within one single UI e.g. between its screens and navigation. However, the attempt to make the mobile UI consistent to the web UI is not as simple as one might expect. This is because of the mobile device’s different UI’s screen size and structure.

In the web UI design, the ability to make all elements visible on the surface already utilizes the usability heuristic minimization of user memory load, as well as clearly marked exits. Furthermore, by following the web navigation conventions, the designer can achieve an increased usable UI. Nevertheless, merely following the web navigation conventions for the mobile UI design does not achieve the same results. The presentation of the entire information on one screen is not applicable for the mobile UI. This problem is illustrated in the two conversion attempts illustrated in figure 23.

\[\text{Figure 23 – Web UI } \leftrightarrow \text{Mobile UI Conversion}\]

\[\text{159 Rhodes et al. 2001, p. chapter 3.1}\]
In the first attempt, conversion 1, all elements are placed in the main screen area causing the coverage of most elements. Only by scrolling down the screen they become accessible to the user.

In the second conversion, soft buttons are used to extend the visible area. But since the lower place is only limited to three soft buttons, the remaining elements will be hidden in the screen’s menu bar. This leads to non-visibility of most elements causing disadvantages such as a higher demand of user’s memory load, invisible exits and screen complexity.

It is obvious that being consistent in the UI design methods does not automatically imply consistency in the new mobile UI. Furthermore, the same utilization of heuristics does not achieve the same usability goals for the mobile UI. Instead, these two conversion attempts would probably make the user get lost in the application, since they decrease the mobile UI’s Memorability, User Experience and Learnability. Taking the results of the described UI conversions into consideration it is clear that a new conversion method must be found.

Since using one screen for the presentation of the entire data is not feasible, it must be considered if a deliberate division into sub-screens in combination with a simple navigation will improve the mobile UI’s usability. For this, navigation elements that were previously integrated into web UI’s screen must be separated and put into one separate screens. The remaining original content must be kept as one screen since it forms one logical unit. The available utilities only comprise the exit and help function. To achieve a simple mobile UI, and furthermore to follow the heuristic of clear exits, these functions must not be hidden in a special form. Instead they can be added to the navigation screen.

The resulting screen structure and navigation follows the defined usability heuristics even if it applies them in a different way than the web UI. Although, the data is located on a different screen a consistency to web UI’s screens is discernible.

Utilizing this conversion method to all screens of the web UI’s that are relevant for the mobile UI results in the screens described in 4.3.3.3. The navigation between these screens is described in 4.3.3.4.

### 4.3.3.3 Screen Layout

For a mobile UI that is most extensive consistent with the web UI, the subsequent screens are designed following the conversion method introduced in 4.3.3.3. The **Main Menu** screen serves as main navigation screen. The **Task List** screen provides the access to the
task instances of mobile UI’s three available task lists (All Tasks, Current Tasks, Completed Tasks). Furthermore, the Task & Task Details screens provide the task instance information. Beside these core screens further screens such as the Synchronization Process, the Login and the Network Setting screen are provided by the mobile UI.

Characteristics of these screens are the strong utilization of the defined usability heuristics, whereby restrictions of two usability heuristics had to be made. First of all, the heuristic to speak the user’s language is only applied to a limited extent since the graphical language is mostly avoided. The reason for this is the difficulty in using icons or pictures to create affordances\(^{160}\). In general, to save resources mobile applications are rather based on a pure textual than on graphical information presentation. Therefore, it was even more important to use a clear and comprehensible textual language in the mobile UI. Secondly, to save memory the usability heuristic of help and documentation has not been utilized. This is acceptable because of the mobile UI’s simplicity and consistency with the web UI. One can expect that the user very quickly and intuitively understands the mobile UI.

- **Main Menu Screen**

The Main Menu is the initial screen the user sees when he launches the application. On the one hand, it serves for screen navigation purposes, and on the other hand it provides major application features. The screen differs in offline and online mode whereby the mode is always made obvious to the user to prevent typical mode errors. This is achieved by explicit naming the specific mode in the title. Furthermore, the user recognizes the mode from the available menu options.

In offline mode the user is enabled to open the login dialog for the connection to PEM’s back-end. In addition, the access to his two local task lists - Current Tasks and Completed Tasks - is provided. Finally, the option to quit the application is placed on the screen following the usability heuristic of clearly marked exits. In addition to the options on the major screen area, the function to access the network setting dialog is hidden behind the title in the upper screen area.

In online mode, all previously described options are adopted. Only the login option is replaced by a logoff option. Beside this, the menu list is

\(^{160}\) Chapter 2.2.2.3 usability heuristic, Speak the User’s language
further extended whereby the access to the task list All Tasks is added to the menu. Furthermore, the function for the initiation of the manual synchronization process becomes available.

The arrangement of frequently used options directly on the screen and rarely applied functions behind the title keeps the screen simple and manageable. Although, the hiding of the network setting function counteracts the usability heuristic to minimize users memory load, the location is justifiable for the following reason: In general the network settings are only performed once in a mobile UI’s lifetime. Furthermore, for this one time the setting dialog can be opened automatically by the UI when the user connects to the PEM engine the first time and the network specifications have not been set.

- **Task List Screen**

This screens major characteristic is the applied consistency heuristic for the display of the mobile UI’s three task lists. The consistency is achieved by using one screen for each defined task list. The task list screen is structured as follows:

Similar to the main menu, this screen shows a menu list for the selection of task instances, whereby the presentation is different from task list of the the web UI. There, the task instances are displayed in a table offering detailed information. However, the advanced handheld’s restrictions of the small screen and the applied menu structure do not allow the display of the same amount of data as in the web UI. If all task details to each task instance were displayed on the screen, the user would only be able to see one task instance at a time. By scrolling down the screen, the other task instances would become visible. This mechanism is not simple to use and does not provide a good overview of the available task instances. Thus, it increases the user’s memory load. Since only one task instance is displayed on the screen, the user must always remember the other task instances. Consequently, to achieve a concise task list, the task details visible on the screen must be kept to a minimum.

The limited task details that will define one task instance must be significant and unambiguous. In addition, these details must be a definition comprehensible to the user and
therefore optimal helping him by his search for the needed task instance. The combination
of process instance ID and task instance ID defines exactly one task instance. But, since
both IDs consist of cryptic numbers they are not meaningful to the user. However, the task
name conveys an adequate meaning to the user, but since several process instances of one
process definition, and therefore, several task instances of one task definition exist, the task
name is not a unique identifier. Instead, it makes sense to use the task name in combination
with another meaningful task detail so that their combination clearly identifies one task.
For this, the composition of task name and the activation date is eligible.

As the task list occupies the main part of the screen, the
available task performing functions must be placed on the
lower part of the screen or in the menu behind the title. The
functions to open a task instance and to exit the screen are
made visible as soft buttons on the screen. But because not
more than three soft buttons can be displayed, the remaining
task functions such as claiming, completing a task are hidden
in the title menu. Although, the coverage of functions should
be avoided to minimize memory load, it is justified for these task functions. This is
because these functions are frequently used, and therefore, the user will very quickly
remember their screen location without any effort.

A further soft button is taken by the exit function to this screen and is always visible to the
user. Finally, to be consistent with the main menu the application mode is written out in the
title.

• **Task: Task & Task Details Screens**

These screens show the same inner consistency as the task list screen. The task instance is
displayed in the same screens independent from the task instance origin of the task lists All
Tasks, Current Tasks or Completed Tasks. However, the task functions always adjust to
the respective task instance’s origin task list.

The separation of the task instance information into two interrelated task screens simplifies
the task handling. However, the two screens, Task and Task Details, differ in their meaning
for the user.

The first (figure 27) contains the static task details such as the process title, the scheduled
start and end date. In the web browser UI, this data is directly included in the task list
pursuing the pure provision of general task information.
The second screen (figure 28) is mainly applied for data collection purposes, since it holds the dynamic, editable task data. For this, input field elements are placed on the screen. Dependent on the task data fields’ format a text field, a number field, a list box, a combo box or a date field is used. The adequate choice of the field element is used to facilitate the data entry and to prevent errors.

The placement of the functions resembles the task list screen’s structure. The major functions comprising the close task instance command and most frequently applied task functions are made visible on the screen; the less frequently used task functions are hidden behind the title. To prevent mode errors, the existing mode name is added to the title.

- **Synchronization Process Screen**

This screen is introduced for the manual synchronization process. While the mobile UI synchronizes the task instance copies of the Current Tasks and Completed Tasks in the background, a progress bar indicator informs the user about the current synchronization progress. However, in case of completion or a premature synchronization abort the system presents feedback about the success or error messages about the synchronization failure. The error messages consist of detailed information about failure reasons enabling the user to perform measures that avoid a failure in the next synchronization process.

Since only the function to exit the screen or to abort the synchronization process is available, they can be made executable by clicking on a soft button placed on the lower screen.

- **Login & Network Setting Screen**

The login screen serves for the authorization process. Here, the user is enabled to type in his user name and password. Additionally, he can initiate the connection to PEM’s engine. The network setting screen allows the user to define and test PEM’s URL, as well as to store this URL permanently on the mobile UI. Both screens are kept very simple and consistent with the other screens. They follow the heuristics of minimization of user memory load, show clearly marked screen exits, prevent errors, and provide adequate feedback and error messages.
4 Concept of a Mobile User Interface

4.3.3.4 Navigation Structure

Whereas in the web UI, the navigation is made visible on the screen, in the mobile UI this is impossible because of the device’s limited screen size. Consequently, it is highly important to equip the mobile UI with a navigation that is easy to remember and simple to use. The navigation must manage the complexity of the multiple screens focusing on the further increase of the system’s learnability, efficiency, memorability, as well as its user experience. The navigation structure that is applied for the mobile UI is the cyclic network belonging to the five navigation structures\textsuperscript{161} defined by Shneiderman\textsuperscript{162}.

In the following, an abstract navigation definition of mobile UI’s core screens is given. The initial screen is the main menu screen. From this screen it is possible to access a task list screen. However the task list screen allows the access to one of the defined task screens, the Task screen or the Task Details screen. As can be seen in figure 29, it is possible to always return to one of the previous screens. For example, from the task screen, the user is enabled to return to the task list or the main menu screen.

The designed screen navigation follows the tenet that the navigation is simple when it does not hold more than three screen levels. Furthermore, the small number of screen levels does not immensely increase the user’s memory load. Giving the user the ability to return to previously visited screens with only one single click raises the mobile UI’s efficiency. In summary the mobile UI’s learnability, memorability and user experience are positively affected by the simplicity of the introduced navigation.

The specification of the abstract navigation definition of figure 29 can be enhanced to a more detailed navigation structure illustrated in figure 30. This detailed navigation definition handles, on the one hand, the different available task lists, and on the other hand, the mobile UI’s two application modes. It still follows the previously explained heuristics and further increases the usability goals that can be influenced by the navigation structure.

\textsuperscript{161} Shneiderman 1997, p.239: (1) Single menu = one single screen; (2) Linear sequence; (3) Tree structure; (4) Acyclic network (5) Cyclic network
\textsuperscript{162} Shneiderman 1997, p.236-252
Figure 30 – Navigation Structure in Detail
5 Prototypical Implementation

This chapter introduces the prototypical implementation to PEM’s mobile UI concept of a thin-smart hybrid running on PDAs and communicators. For this, the platform considerations that had to be made, as well as the core implementation are illustrated. Furthermore, the known limitations of this prototype are presented.

5.1 Platform Considerations

As can be seen in figure 31, the existing system architecture has been extended by a mobile user interface. Furthermore, a back-end component has been added to serve as the interface between PEM’s BPM engine and mobile client. To find the correct combination of underlying software components needed by the mobile UI, it is necessary to examine the different possibilities suitable for a mobile UI.

![Figure 31 – Extended System Architecture](image)

5.1.1 Front-End Software Decisions

For the front-end client, the application type, the data storage type, as well as the device’s operating system must be defined.

5.1.1.1 J2ME Application as Client

A smart client can be realized either as a native application, using native programming languages such as C or C++, or as a Java application when implemented with the cross-platform programming language Java.\(^{163}\)

\(^{163}\) Mallick 2003, p.196
Native applications usually show a better performance than applications running in virtual machines such as Java. The reason for this is that native applications can be optimized for the specific device and/or platform they are to run on. However, Java applications require an additional abstraction layer, the Java Runtime Environment (JRE). In comparison to native applications Java applications hold the immense benefit of cross-platform capabilities. They are executable on every operating system (OS) for which a JRE is available. Furthermore, Java is known as a very powerful programming language. Its pure object model without any pointer mechanisms reduces errors and increases productivity.

However, the decision for the implementation of a Java client application is mainly based on Java’s platform independence. Thus, in further developments it is possible to simply adjust the prototype from one OS to another. Furthermore, it is decisive that TRAXION’s BPM engine class library is also implemented in Java. Applying Java on the front-end, as well as on the back-end application will allow the reuse of source code. The lower performance of Java applications can be accepted since mobile devices will continuously improve their processor power. It is conceivable that in the near future all standard advanced handhelds will possess the needed processor power for a fast mobile UI.

- **Java 2 Platform, Micro Edition (J2ME)**

J2ME is a platform that is primarily developed for devices with limited memory, display and processing power such as handhelds and embedded devices. Tailored to these devices, it represents a subset of the Java 2 Platform, Standard Edition (J2SE) API that is enhanced with additional device specific APIs. Since J2ME covers a broad range of diverse devices, it is inexpedient to provide equal APIs independent of the specific device. Consequently, to save memory space the Java platform is divided into several parts that can be put together as necessary. These parts are categorized into configurations and profiles. A configuration represents the minimum platform for its target device. A profile is a higher-level API that further defines the application lifecycle, the UI and the access to device specific properties. The configuration that is used for the implementation of the new mobile UI is the Connected, Limited Device Configuration.
(CLDC). It is based on a small JVM, Sun’s K Virtual Machine (KVM) and is designed for devices with interrupted Internet access, extremely slow processors and limited memory of 128 kilobytes. However, the chosen profile is the Mobile Information Device Profile (MIDP). It offers functionality such as the UI for handhelds, network connectivity, local data storage, and application management\textsuperscript{171}.

5.1.1.2 J2ME Record Store for the Data Storage

The options that are available for the persistent storage of data on an advanced handheld are proprietary databases, commercial relational databases, and custom-coded databases\textsuperscript{172}. Proprietary databases are provided by the selected mobile OS, as well as by the J2ME platform. In most cases these databases are flat-file databases storing data as single sets of records. This database structure leads to limited feature and storage capacity e.g. the restriction to one record format within the database, or the lack of record search by column value. Furthermore, the management of these databases must be implemented into the client application. However, in general they are characterized by a quick and low-cost application development. While, proprietary databases are suited for applications with small data storage, applications with large persistent data volume will suffer in their performance. Most mobile operating systems such as Palm OS and Windows mobile, as well as J2ME provide typical flat-databases possessing the described characteristics. But they differ in their platform independence. Because of J2ME’s nature, its database runs on diverse operating systems, whereas the mobile databases of Palm OS and Windows mobile are restricted to their own operating systems.

Commercial mobile relational databases hold the table structure of columns and rows. Therefore, it is possible to store content of different data types. Moreover, they provide enhanced features for the access and the management of data. One of these features is the server database synchronization capability, another is the fast filter function. These databases are particularly well suited for applications with complex and high data volumes. Another essential characteristic is their platform independence. The deployment of independent databases always implies the need for additional memory storage. Moreover, by using a commercial database the initial costs and the dependence on this software must be considered. Examples for mobile commercial relational databases are Sybase SQL Anywhere Studio, DB2 Everyplace, Oracle9i Lite and SQL Server CE.

\textsuperscript{171} Sun 2002 b
\textsuperscript{172} Mallick 2003, p.199-218
In comparison to the proprietary and the commercial database, custom-coded databases possess the advantage of being optimized for the specific application and OS. But they also always imply immense development efforts.

J2ME’s proprietary database is chosen for various reasons. The first reason is a possible future enhancement to enable a broad range of devices to apply the mobile UI. This already eliminates OS specific proprietary databases. Comparing J2ME’s proprietary database and commercial relational databases, the J2ME’s database is more appealing. Since the tasks needed to be stored on the mobile device can be limited to a small number, no complex database functions are needed. Furthermore, with this solution the prototype remains platform and third-party software independent. The implementation of a customized database solution is discarded because it does not imply significant value for the mobile UI that would justify the high development efforts.

5.1.1.3 Palm OS Device as Mobile Device

In the mobile operating system market, several operating systems are present\(^{173}\). Current mobile operating systems include Windows Mobile, Palm OS, Symbian and Linux. The OS that is mainly represented in the USA - CommerceQuest’s current major sales market - are Windows mobile and Palm OS\(^{174}\). In the following, these OS are compared with each other creating a basis to choose one OS for PEM’s mobile UI prototype.

Windows Mobile and Palm OS have two entirely different conceptual ideas and development histories. Palm OS pursues the concept of not adjusting a PC to fit into a smaller form factor. Instead, the Palm OS is tailored to the users’ needs providing a small set of adequate, simple functions\(^{175}\). However, Windows Mobile can be described as miniaturized version of Windows\(^{176}\) focusing on the cooperation and consistency between these two OS. Since Palm OS and Windows mobile are in strong competition with each other, they constantly enhance their features thereby becoming increasingly alike.

With a mere consideration of functionalities it is difficult to determine one preferable OS as basis for PEM’s mobile OS. Therefore, additional decision criteria such as their market share must be considered. Looking at the market penetration, Palm powered devices dominated in 2002 with 80 percent share in the USA\(^{177}\). Despite this fact, Windows mobile

173 Hubley et al. 2004
174 Mallick 2003, p.159
175 Hayes 2002, p. 71
176 Weiss 2002, p.36
177 PalmInfocenter 2003
5 Prototypical Implementation

is the OS that is most applied for corporate mobile device deployment\textsuperscript{178}. Since the PEM is developed for business use, it is reasonable to provide a mobile UI as final product that uses Windows mobile as operating system. However, the OS that is chosen for the prototype is Palm OS. The reason for this is the existing development environments for Palm OS – J2ME software. In these development environments, the compiled source code can be tested on a Palm OS emulator running on the PC. However, Windows mobile – J2ME software always must be tested directly on the mobile device. Using an emulator for test purposes implies a more cost-efficient solution than testing on real devices. Although the Palm OS is selected as current OS, for the further development decisions the operating system windows mobile is also taken into consideration. This is because the Windows mobile is a highly interesting OS for possible future mobile UI enhancements.

5.1.2 Java Servlet as Back-End Interface

The finding of software components for mobile UI’s front-end is extensive and complex, but the same does not apply for mobile UI’s back-end interface. Here, a Java Servlet is the only possible solution. A Java servlet is the Java platform technology that provides mechanisms to enhance web servers’ functionality and to access existing business systems\textsuperscript{179}. However, one reason for the choice of a Java servlet is that the BPM engine class library provides almost entirely the functionality necessary for the mobile UI comprising the functions to access and perform task instances. Therefore, a direct synchronization between the back-end database and the mobile front-end database using e.g. SyncML would only result in the needless creation of business logic that already exists\textsuperscript{180}. Consequently, a component must be integrated into PEM’s BPM engine that handles the communication between the BPM engine and the mobile UI. Since the BPM engine is implemented in Java, it is reasonable to also create the interface in Java. But in the end, the decisive reason is that a J2ME application can only use a Java Servlet when the back-end interface is realized as a Java component.

5.2 Core Implementation

Topics of this chapter are the mobile client’s inner structure, its components’ behaviour and their relationships, as well as the mobile servlet’s characteristics.

\textsuperscript{178} Mallick 2003, p.195
\textsuperscript{179} Sun 2004 b
\textsuperscript{180} Mallick 2003, p.247-253
5 Prototypical Implementation

5.2.1 Mobile Client's Components

PEM’s mobile client is developed as MIDlet. A MIDlet is a J2ME application executable on those devices that support the MIDP [181]. For the mobile client implementation, the developed MIDlet consists of several components representing either one single Java class or a group of logically interrelated Java classes. The relation and the interaction between these components are topic of this chapter.

- **MIDlet**

  The basic component of the mobile client’s inner structure is an instance of J2ME’s abstract MIDlet class. Launching the mobile UI instantiates this component, and it will remain in the background during the application’s entire lifetime. The MIDlet component provides general functions to set the application into the J2ME MIDlet system states comprising active, paused and destroyed. They are either invoked by the mobile client’s remaining components described below, or by the mobile device itself. However, the MIDlet component of PEM’s mobile client mainly manages the setting of the system state *started*. For this, it initiates the processing of the locally stored data, the provision of the first screen, as well as the screen management. Furthermore, it defines the variables that must exist from the start until the exit of this application, or variables that must be accessible from every component. The MIDlet component relates with the *Displays* component, the *Display Manager* component and the *Record Store* component.

- **Displays**

  The screens defined in the mobile UI’s physical design are created as displays in the prototypical implementation, whereby for each screen exactly one display exists. Essential for their design is the main focus on the strict separation of business logic and presentation information. Therefore, the displays almost exclusively consist of presentation information, whereas the business logic is encapsulated in other components such as the Task Handler component. Consequently, for the utilization of business logic, the displays continuously interact with the components that are equipped with the display’s intelligence. Finally it is to mention that the displays not only hold the presentation information, but they also define the forward navigation to each of the following screens. Beside these general characteristics, each display has it’s own specific functionality.

  The display of the *Main Menu* is rather simply designed. Here, a list element is used to

---

[181] Topley 2002, chapter 3.3
present static list items. These list items enable the access to the task lists; Current Tasks, Completed Tasks, All Tasks, as well as the exit of the application. Furthermore, they enable the login, logoff or synchronization procedure depending on the current application mode. The change from one mode to another adds or removes possible list items.

Similar to the main menu display the Task List Display is also created with a list element, whereby here a list item label represents one task instance. With every new instantiation, list items are created dynamically depending on the different application states online or offline mode, as well as the variant task list types Current Tasks, Completed Tasks, All Tasks. Therefore, the task instance is either taken from the local record store or PEM’s back-end database. For this, the task list display interacts with the task handler component requesting the data from the record store component or the HTTP Interface component. The available task functions such as the functions to claim, save, complete or release a task are defined in the task list handler component. Depending on the application’s mode, the task list handler component informs the task list display about the functions that must be provided to the user.

The Task Details Display and the Task Display\textsuperscript{182} use a form element instead of a list element for the presentation of the task information belonging to one task instance. One form element is applied for each display. Whereas, the task detail’s display presents the task instance’s task details, the task display shows the task instance’s task data. For the presentation of this task information field elements are placed on each display. Whereas, the task details are exclusively viewed in non-editable text elements, the task data is put into field elements editable for the user. These editable field elements are of the type text, number, date, combo box or list box. Moreover, the arrangement of the task details field elements onto the task details form element is constant and defined in the mobile client. However, the arrangement of task details field elements on the task details form element, their number and their type differ between each task instance. This is because the task data field elements are never equal since with every new process definition the task data differs. Thus, it is not possible to define the task data field elements once for all task instances. Instead, for each task instance they must be automatically defined and arranged on the form element conforming to the existing task detail’s field sequence, number and type deriving from PEM’s back-end database. Finally, to gain the display’s business logic, the two displays interact with the task handler component. It informs them about the task functions these displays must provide, as well as the function’s behavior depending on the

\textsuperscript{182} Chapter 4.3.3.3
5 Prototypical Implementation

existing application mode online or offline. Furthermore, this component provides the displays with the task information, which will be presented in the field elements.

Beside the previously described displays, further displays for the Data Synchronization screen, the Login screen and the Filter Setting screen exist. These displays are characterized by their constant screen structure, as well as the functionality provided. They interact with the HTTP Interface component, as well as with the task and task list handler component.

- **Display Manager**

As described in the mobile UI concept, the UI holds a complex cyclic screen navigation structure. It allows the forward navigation from one display to the subsequent displays. Furthermore, it enables a backward navigation to the already visited displays. The static information for the forward navigation is stored in each display. However, the same storage location is rather inefficient for the dynamic backward navigation information. Instead, to keep the application logic simple and fast, it is more reasonable to store this information in an extra component that remembers in which sequence the displays are visited. For this, the display manager is created using a stack structure for the storage. It provides functions to store the display references, as well as functions to access them in the sequence they occurred.

- **Task & Task List Handler**

As mentioned before, these components are used by most of the defined displays providing them with business logic and task information. Whereas, the task handler handles the task information of one task instance, as well as its functions, the task list handler manages all task instances of the task lists; Current Tasks, Completed Tasks or All Tasks. For this, the task list handler applies the task handler’s logic. Since both components behave similarly, a good understanding can be gained from the detailed introduction of only one component. Thus, the task handler component is chosen to illustrate the nature of these components, as well as their interaction with other components comprising the Displays component, the HTTP Interface component, the record store component or the mobile Task component.

When the task display is first initiated, it requests its data using the task handler component whereupon this component checks the task type and, depending on this type, chooses the further action necessary. In the case where the task instance belongs to the task list All Tasks, the task handler starts a communication with the HTTP Interface component receiving and sending the task information from and to PEM’s back-end. Additionally, if
5 Prototypical Implementation

the user downloads and saves a task instance copy on the mobile device another communication to the local record store is initiated. In a second case, the task instance belongs to the task list Current Tasks or the task list Completed Tasks and the mobile client is in offline mode. For this type of task, the task handler starts only one communication with the local record store component. The final case occurs where the task instance belongs to the Current Tasks and the mobile client is in online mode. Then, the task handler component communicates with the record store component and possibly with the HTTP Interface component to enable a direct update with the task instance originals on the server.

![Component: Task Handler](image)

However, the task list handler component is applied by the task list display in the same manner as the task handler component by the task display. Furthermore, the task list component is also accessed by the synchronization process display. Here, the task list handler component opens a connection to the HTTP Interface component, as well as to the local record store component. It initiates the update of Current Tasks' task instance copies. Furthermore, it sends the Completed Tasks’ task instance copies to the PEM engine initiating the completion process of their original task instances.

Since the record store component, HTTP Interface component and the display component work with the task information in different formats, the task information must always be converted to the required format. For this, the task handler component makes use of the mobile task component.

- Mobile Task

The mobile task component serves as intermediate storage location for one task instance temporarily saving its task details, as well as its task data\(^{183}\). This information is loaded into the mobile task component when the task instance is presented in a task display.

---

\(^{183}\) Chapter 3
before it is permanently stored in the J2ME record store or before it is sent to PEM’s engine. In each of these cases, the task information is converted into the correct format. However, to save memory space the task instance information only comprises a subset of the task information on the server. For the distinct definition of the original task instance belonging to its downloaded copy, as well as the copy’s own identification data, it is necessary to save the process instance ID, the task instance ID and the date of the last synchronization with the server. Furthermore, the task details and task data that is relevant for the user to process a task instance is saved in the mobile task component.

- **Record Stores**

In the mobile UI’s physical design concept, the local task instance copies are allocated to two task lists, the Current Tasks and the Completed Tasks. This division of task instance copies is also adapted for the physical local data storage. Thus, two record stores must be created, the current tasks record store and the completed tasks record store. Although, the data structure of these two databases is exactly the same, for performance purposes it is essential to physically separate these task instance copies. This can be explained as follows. In the mobile UI the two different task instance types are always presented independently. Therefore, the provision of only one single record store for both types requires a filter setting before every task list presentation. But, applying a filter in a flat file system database such as J2ME’s record store demands a lot of the mobile device’s limited processor power. This is because the filter setting is not conducted on one specific column but on the entire task instance record. Therefore, every record must be entirely read, interpreted by the system and then checked if it belongs to the filter. However, the creation of two databases eliminates this complex filter procedure, and furthermore requires less processor power by the mobile client.

The two record store components provide the functions to read and write existing records, to add new records, as well as functions for the search for a specific record. To prevent errors the task information must be stored in a constant sequence of task information fields. The task information’s format in the record store is the byte array.

- **HTTP Interface**

The HTTP Interface component possesses the functionality to initiate an HTTP session with PEM’s back-end through the communication with its mobile servlet. Furthermore, it provides the methods for the generation and the sending of HTTP requests are used by the

---

184 Chapter 4.3.2.2
5 Prototypical Implementation

task and the task list handler component. They enable these components to claim, save, complete and release a task instance on the server by processing the requests to the mobile servlet. Finally, the HTTP interface component receives HTTP responses from PEM’s back-end. It interprets and converts these responses into a comprehensible format that is passed back to the component that previously performed the request.

5.2.2 Back-End Extension

For the integration of an additional mobile client, it was necessary to extend the PEM’s back-end with a Java servlet. Moreover, some smaller adjustments in the BPM engine had to be made.

- Mobile UI’s Servlet

The first feature of this servlet is the management of the user’s authorization. For this, it requests the user’s permission for the PEM access by the BPM engine. Where the permission is given, the servlet creates a new session and sends its session ID back to the mobile UI. Afterwards, with every further request the mobile UI must submit this session ID. It is needed for the continuous verification of the user’s access authorization. However, in case of a rejection, the servlet generates a response that it sends to the mobile client to inform the user about the failed system access.

Furthermore, the provision of the adequate task instances is also a feature of this servlet. It instantiates a filter setting on the task instances stored on PEM’s back-end database leaving only those task instances that are relevant for the mobile editor. The task information of each task instance is further reduced to a minimum that is absolutely necessary for the task instance presentation or storage of the mobile client. The preparation of only the actually needed task information keeps the transfer volume small, and therefore decreases the mobile UI’s response time for the communication with the BPM engine.

Beside the provision of data, the servlet also handles the mobile UI’s task functions comprising the functions to claim, save, release and complete a task instance. For this, the servlet checks if the user is allowed to change and update the original task instance on the server. If no synchronization conflicts occur, it converts the task information sent by the mobile UI into BPM engine’s required format. After that, the servlet passes this information to the BPM engine. In case of a conflict the change of the original task instance on the server has a higher priority than the task instance copies on the mobile client. Consequently, the servlet must generate and return a response that informs the
Prototypical Implementation

mobile client about its detected conflict. With this information the mobile client can initiate the further conflict resolution process on the mobile device.

- Additional Adjustments to PEM’s Back-end

The PEM’s system architecture shows a strong integration of the original web UI into PEM’s back-end. Consequently, some of the classes and functions provided by the PEM’s BPM class library are designed to mainly support the interaction with this web UI. However, for the interaction with the mobile UI it was necessary to adjust some of these functions’ source code, as well as to add some minor functions. Since these enhancements don’t change BPM’s class library core functionality but simply enable the mobile UI’s integration, they are explained in no further detail.

As well as these changes of the PEM’s source code, adjustments were made in the back-end’s static application data. The introduction of the new system user mobile editor requires detailed specifications of this user’s data access rights. The specifications are stored in the PEM’s back-end database.

Finally, the application must be informed about the new servlet and its location. This is done in the application server’s configuration.

5.3 Known Limitations

The deployment of mobile UI’s prototypical implementation shows one major limitation.

With regard to J2ME specifications, MIDlets should be executable on every device with the Palm OS 3.5 or higher. However, when testing the mobile UI on different devices, this has been shown to not always hold true. The mobile UI ran without any problems on tested devices with Palm OS 5, whereas problems occurred on devices with the Palm OS 3.5. Very often, either the mobile client was very slow, or it was not entirely executable. The reason for this is that devices featuring the Palm OS 3.5 are rather older devices with low processor power and very limited memory. In general, one can say that the mobile client runs very well on the currently available devices, but shows performance problems on older devices. However, for each device the mobile client must be tested directly on the device before it can be deployed.

---

185 Sun 2002 c
6 Outlook

The previous chapter describes the prototypical implementation to PEM’s mobile UI concept. It is developed as J2ME MIDlet executable on PDAs or communicators featuring the Palm OS 3.5 or higher. The main focus in the design process has been the assurance of the defined usability goals. However, due to the prototypical nature, the mobile UI offers potentials for the further increase of its usability. In addition, the prototype can be enhanced for its deployment in alternative scenarios.

6.1 Possibilities for Further Development

The first potential for enhancement stems from the generic arrangement of task details field elements on the task details form element.\textsuperscript{186} As said before, the sequence of field elements for this arrangement is derived directly from PEM’s back-end database. There, the task field elements are stored in alphabetical order. Consequently, the field elements are also arranged in an alphabetical order on the task details display. This alphabetical order complicates the data entry, and therefore negatively affects particularly the usability goal User Experience. Instead of an alphabetical order, an order depending on the task field elements logical coherence would be more suitable since it is easier to read. For example, comparing the alphabetical order of contact data (city, name, street, surname, zip code) to these fields’ logical coherent order (surname, name, street, city, zip code) clearly illustrates the issue. However, the automatically logical arrangement of task details elements cannot be achieved through adjustments of PEM’s back-end or of the mobile client. Instead, TRAXION’s software for the creation and the design of process definitions the Process Execution Manager must be enhanced. Here, it must be made possible to define the field elements sequence for their presentation on the display.

However, another possibility for mobile UI’s improvement aims at a higher efficiency and user experience for the task list All Tasks. Whereas, the local task lists Current Tasks and Completed Tasks is restricted to ten task instances, All Tasks allows the remote access to the entire Mobile Tasks on the server. The number of these task instances differs depending on the number and complexity of the process instances managed by the PEM. Consequently, the number of task instances can be extremely high making it difficult for the user to find the required task instance. This problem can be counteracted by the introduction of filter mechanisms for the user. For this a new screen must be designed that enables the user to define certain filter criteria such as the process name, task name,

\textsuperscript{186} Chapter 5.2.1.2
activation date, etc. Furthermore, the task list handler component, as well as the mobile servlet must be adjusted to handle the user’s filter requests.

Another increase in the system’s user experience can be accomplished by the ability to set user preferences. The ability to define each user’s personal order of task instances within the task lists is only one example. Furthermore, the personalized definition of the task list’s item label is also very helpful. Instead of the currently defined default label consisting of the task name and the activation date, the user creates his personal item label. For this, he uses task details elements that are particularly meaningful for him. However, for the enabling of this preference setting feature a screen must be implemented that allows the user to specify his preferences. Furthermore, an additional record store to provide the preferences’ permanent storage ability must be created. Finally, a few adjustments must be made in the mobile UI’s source code.

Beside the extensions of mobile UI’s features, the system’s usability can be further improved by making it available for the system user supervisor. Similar to the conversion of the system user editor into the system user mobile editor, a specified supervisor that is the mobile supervisor must be created\textsuperscript{187}. For this, the supervisor’s data and functions are adjusted to the mobile context. Adding the system user mobile supervisor allows to use the mobile UI, for example, for the deletion of a task instance or for delegation of a task instance to a different user.

Finally, as already mentioned before, it is reasonable to extend the mobile client for a deployment on the operating system Windows mobile\textsuperscript{188}. Although, J2ME applications are defined as platform independent it is probably not possible to apply the mobile client without making any adjustments. This is because the J2ME behaves differently on different mobile devices and operating systems since it always adjusts to the specific conditions. However, the extent and nature of the necessary changes requires a more detailed investigation.

### 6.2 Deployment in Extended Scenarios

The PEM’s mobile UI is designed to support the user in the processing of his mobile tasks, whereby the data entry is performed manually. However, for some tasks it is reasonable to automate this manual procedure to increase the process’s efficiency. These tasks are characterized by a high data volume and well-structured data format that is somehow

\textsuperscript{187} Chapter 4.3.1.2
\textsuperscript{188} Chapter 5.1.1.2
comprehensible by the mobile device or by another device connected to the mobile device. One example for this data is the bar code data. For the automatic bar code entry and task processing, the mobile device must be enhanced by a bar code scanner add-on. Furthermore, the mobile client must provide the feature to enable the communication with the bar code scanner, as well as the feature to automatically enter the bar code data into the task. However, for each data format that is automatically entered into the mobile UI, the mobile UI must be enhanced by a communication and data input feature specified for the respective data format and the device that is applied for the data entry.
7 Summary

In the previous chapters, an overview of the concept of Business Process Management (BPM) has been given and different approaches to the optimization of business processes have been explained. Furthermore, the efficiency potential of software support in both the (re-)design of business processes, as well as the repetitive execution of these business processes has been identified. The requirements the BPM software must fulfill to fully leverage these potentials and the different core capabilities of such a software system have been briefly explained. With all software support it becomes clear that business processes will still be executed with extensive human work, thus the extent of how much of the efficiency potential any BPM software will be able to realize will be dependent on the efficiency of its user interface (UI), among other things.

The design of user interfaces is the subject of the discipline of Human-Computer Interaction (HCI). Thus subsequently an overview over the fundamental concepts of HCI, UIs, the idea of Usability and finally the engineering approach to designing human-computer interaction interfaces following a formalized method are explained in detail, especially the usability goals and usability heuristics.

In the following, an exemplary web-based Business Process Management system, TRAXION by CommerceQuest, is analyzed with respect to the previously stated usability heuristics. To do so, at first, an overview over the general architecture and conceptual design of TRAXION is given. Particular attention is, of course, paid to the web-based user interface and how the usability heuristics are applied and implemented.

Next, an analysis is given of how a BPM software in general and TRAXION in particular can extend its possible usages and thus extend its potential customer base. Typical requirements for users in business processes are identified and limitations of TRAXION in supporting mobile users in non-office environments are identified. These limitations are mainly due to the exclusive web-based user interface which limits the usage of TRAXION to situations where the user has an internet-connected PC at hand. Typical mobile users are classified and their requirements, especially regarding the user interface, are derived, resulting in the suggestion that a mobile user interface to BPM-systems such as TRAXION would create a competitive advantage.

Subsequently the conceptual design of a mobile user interface to a BPM software is created by systematically applying the previously defined usability goals. Multiple current mobile device technologies as well as multiple client software architectures are
analyzed to establish how far they meet the usability goals, leading to the ideal combination of a “hybrid client” (i.e. a mix of thin and smart client) software architecture running on “advanced handheld” devices. After having defined the target platform and general architecture in the conceptual design; the specific components and their functionalities as well as their realization with a mobile user interface are defined, using the usability heuristics to ensure the optimal fulfillment of usability goals.

Finally, the prototypical implementation of the design is described. The necessary decisions for a concrete device, operating system as well as programming language and run-time environment are explained as well as the modules of the newly developed software and its integration interface to the back-end system of the TRAXION BPM software.

In conclusion, a view over the future is given, identifying potential further development of the described concept and software as well as their extension to further usage scenarios.

Thus the questions initially posed in the introduction are answered and the suggested results are delivered. Typical characteristics of mobile users have been identified and their requirements have been classified and specified. Furthermore, the structured method of human-computer interaction design has been identified and adapted to be appropriate for the design of mobile user interfaces to BPM software systems. Finally this adapted method has been applied to develop a prototype of a mobile UI to the TRAXION BPM-system, which meets the previously identified requirements.
8 References

Barnes 2001


Berkun 1999


Berkun 2000 a


Berkun 2000 b


Beaulieu 2001


Burlton 2001


Brans 2002


Carroll 2001


CommerceQuest 2003a

8 References

CommerceQuest 2003 b
CommerceQuest: *Process Execution Manager - Documentation*, Version d7.2.0.0, 2003

CommerceQuest 2004 a
CommerceQuest: *Introducing CommerceQuest – Press Release*,

CommerceQuest 2004b
CommerceQuest: *Process Execution Manager – Press Release*,

Cooper 1995

Fischer et al. 2002
Fischer, Jochen; Herold, Werner; Dangelmaier, Wilhelm; Nastansky, Ludwig; Suhl, Leena: *Bausteine der Wirtschaftsinformatik – Grundlagen, Anwendungen, PC-Praxis*, Erich Schmidt Verlag, 2002

Davenport 1993

Donahue 2001

Evans 2001

Ficher 2004
8 References

Funk 2003
Funk, Jeffrey: Mobile Disruption – The Technologies and Applications driving the Mobile Internet, Wiley-Interscience, 2003

Hackos et al. 1998

Harmon 2002

Harrington et al. 1997

Hayes 2002

Hewett et al. 1992
Hewett, Thomas T.; Baecker, Ronald; Card, Stuart; Carey, Tom; Gasen, Jean; Mantei, Marilyn; Perlman, Gary; Strong, Gary; Verplank, William: Acm Sigchi Curricula for Human-Computer Interaction, The Association for Computing Machinery – Chapter 2: Human Computer Interaction, New York, 1992, http://sigchi.org/cdg/cdg2.html, 04/13/2004

Hubley et al. 2004
Hubley I. Mary; Troni, Federica Troni; Kort Todd; Clark, William: PDA and Smartphone Operating Systems: Technology Overview, Gartner, 2004

Hyperdictionary 2004

IBM 2004 a

IBM 2004 b
8 References

IBM 2004 c

IBM 2004 d

Jacka et al. 2001

Khodawandi et al. 2004
Khodawandi, Darius; Pousttchi, Key; Winnewisser, Christian; Mobile Technologie braucht neue Geschäftsprozesse, Universität Augsburg, Lehrstuhl für Wirtschaftsinformatik und Systems Engineering, 2004

Knudsen 2003

Krug 2000

Lehner 2003

Linthicum 1999
Linthicum, David S.: Enterprise Application Integration(Addison-Wesley Information Technology Series), 1st edition, Addison-Wesley, 1999

Mallick 2003

Microsoft 2004

Muchow 2001
Muchow, John W.: Core J2ME Technology – Technology & MIDP, 1st edition,
8 References

Prentice Hall PTR, 2001

Nielsen 1994

OMG 2003

Ould 1995

Pall 1999

PalmInfocenter 2003

Preece et al. 2002
Preece, Jennifer; Rogers, Yvonne; Sharp Helen: *Interaction Design – Beyond Human-Computer Interaction*, John Wiley & Sons, New York, 2002

Raskin 2000

Rhodes et al. 2001

Rosson et al. 2001

Shneiderman 1997
Shneiderman, Ben: *Designing the User Interface – Strategies for Effective Human-
8 References


**Sinz et al. 2000**


**Sun 2002**


**Sun 2002 b**


**Sun 2002 c**

Sun Microsystems: *MIDP for Palm OS 1.0 - Developing Java Applications for Palm OS devices*, http://developers.sun.com/techtopics/mobility/midp/articles/palm, 04/14/2004

**Sun 2003**


**Sun 2004**


**Sun 2004 b**


**Smith et al. 2003**


**Turowski et al. 2004**

Turowski, Klaus; Pousttchi, Key: *Mobile Commerce – Grundlagen und Techniken*, Springer, 2004
8 References

Topley 2002

Weiss 2002

W3C 1999
World Wide Web Consortium: *Hypertext Transfer Protocol – HTTP/1.1*,
http://www.w3c.org/Protocols/rfc2616/rfc2616.html, 04/13/2004

W3C 2004
World Wide Web Consortium: *Cascading Style Sheets*,
http://www.w3.org/Style/CSS/, 04/13/2004
Appendix A  Structure & Utilization of the CD

The CD attached to this thesis consists of the document at hand, in PDF and MS Word format. In addition, the online literature that was used for the preparation and the paper’s elaboration is enclosed. To convey a further understanding of the implemented mobile UI, end user information about the prototype, instructions for its deployment, as well as software components for testing the prototype itself on a Windows-PC are included. The prototype normally depends on an application server serving the business process tasks, which can’t be included in this thesis due to license restrictions. Thus, the prototype of the mobile UI included on the CD contains hard-coded sample data, which can be manipulated as if they were downloaded from an application server. Finally, the API, the source code and the compiled class files are stored on the CD.

Figure 33 – CD Interface

The CD contains an HTML-based navigation UI, which is automatically launched, when the CD is inserted into an autostart-enabled CD-Drive on a Windows-PC. On any other platform or with the autostart feature disabled, the navigation UI can be used by manually opening the “index.html” file in any web-browser.

The navigation UI is composed of a navigation structure located on the left area, as well
Appendix A  Structure & Utilization of the CD

as the content presentation on the UI’s main area. The navigation structure is categorized into two topics labeled “Diploma Thesis” and “Prototype”. Under “Diploma Thesis” the written elaboration of this thesis together with the referenced online literature can be found. Under “Prototype” the prototype with additional documentation such as the official press release and a Macromedia Flash demo of PEM’s mobile UI, the Palm emulator software, Sun’s kVM, the prototype as a ready-to-install Palm application (in PRC format) as well as the compiled Java class files and the Java source code are included.
Appendix B  Vendor List

The following table lists the software products (and vendors respectively) used in the prototypical implementation of PEM’s mobile UI developed in this thesis:

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Software Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Software Foundation</td>
<td>Apache HTTP Server</td>
</tr>
<tr>
<td>Apache Software Foundation</td>
<td>Tomcat</td>
</tr>
<tr>
<td>CommerceQuest Inc.</td>
<td>Process Execution Manager</td>
</tr>
<tr>
<td>CommerceQuest Inc.</td>
<td>Process Flow Modeler</td>
</tr>
<tr>
<td>CommerceQuest Inc.</td>
<td>Process Resource Modeler</td>
</tr>
<tr>
<td>Eclipse Foundation</td>
<td>Eclipse</td>
</tr>
<tr>
<td>IBM – International Business Machines Corp.</td>
<td>DB2</td>
</tr>
<tr>
<td>IBM – International Business Machines Corp.</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>Internet Explorer</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>SQL Server</td>
</tr>
<tr>
<td>PalmOne Inc.</td>
<td>PalmOS</td>
</tr>
<tr>
<td>PalmOne Inc.</td>
<td>PalmOS Emulator</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>Java 2, Enterprise Edition</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>Java 2, Mobile Edition</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>J2ME Wireless Toolkit</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>K Virtual Machine</td>
</tr>
</tbody>
</table>

Table 4 – Software Products used in the Prototypical Implementation
Furthermore, the following table lists the software products and vendors mentioned additionally in this thesis:

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Software Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM – International Business Machines Corp.</td>
<td>DB2 Everyplace</td>
</tr>
<tr>
<td>Different distributors</td>
<td>Linux</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>Internet Information Server</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>SQL Server CE</td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>Windows Mobile</td>
</tr>
<tr>
<td>Netscape Communications Corp.</td>
<td>Netscape Navigator</td>
</tr>
<tr>
<td>Oracle Corporation</td>
<td>Oracle 9i Lite</td>
</tr>
<tr>
<td>Sybase Inc.</td>
<td>SQL Anywhere Studio</td>
</tr>
<tr>
<td>Symbian Ltd.</td>
<td>Symbian OS</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>Java 2, Standard Edition</td>
</tr>
</tbody>
</table>

Table 5 – Software Products mentioned in this Thesis