

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**An Interaction Constraints Model for
Mobile and Wearable Computer-Aided Engineering Systems in
Industrial Applications**

Christian Bürgy

Diplom-Bauingenieur,
Technical University of Darmstadt, Germany, 1999

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Committee Members:

Professor James H. Garrett, Jr., CEE, Carnegie Mellon University

Professor Burcu Akinci, CEE, Carnegie Mellon University

Professor Dan Siewiorek, HCII, ECE, SCS, Carnegie Mellon University

Dr. Amin Hammad, Dept. of Information Science and Telecommunications,
University of Pittsburgh

Department of Civil and Environmental Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

May 2002

UMI Number: 3045237

UMI[®]

UMI Microform 3045237

Copyright 2002 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

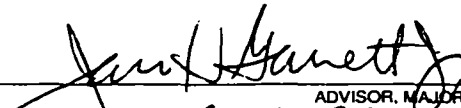
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy


TITLE An Interaction Constraints Model for Mobile and
Wearable Computer-Aided Engineering Systems in
Industrial Applications

PRESENTED BY Christian Bürgy

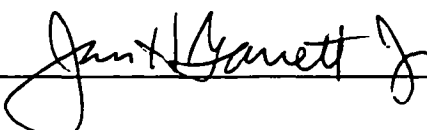
ACCEPTED BY THE DEPARTMENT OF
Civil and Environmental Engineering


ADVISOR, MAJOR PROFESSOR

5-10-02
DATE


DEPARTMENT HEAD

5-12-02
DATE

APPROVED BY THE COLLEGE COUNCIL

DEAN

5-16-02
DATE

Abstract

This research illustrates an approach to improve the design process of mobile and wearable computer-aided engineering systems (*m/w-CAE Systems*). Domain experts, i.e., system designers whose background is in the domain in which the system will be used, rather than in system engineering or software development, need support during the user interaction design process. The developed *Interaction Constraints Model* proves to be a practical approach to meeting this need. Through decision support systems based on the *Interaction Constraints Model*, we can facilitate and speed up the design process for *m/w-CAE Systems* that are used in industrial environments.

The developed *Interaction Constraints Model* maps constraints of specific situations in which mobile IT support is needed to user interface components that may be incorporated in the system design. Due to the nature of industrial applications, these situations mostly are work situations, i.e., situations in which users of mobile and wearable computers work at a specific location of the worksite and have to perform an actual job. This means that the user's interaction with the device is not only constrained by the physical location, but also by the activities that are supported by the device. The importance of location and activity evolved from the opportunity to establish IT support at the actual workplace through *m/w-CAE Systems*. The fact that the computer support moved from a central location, such as the desktop or a kiosk-like computer, to "anywhere" on the worksite makes it inevitable during the design process to take into account the location of the mobile worker. The fact that mobile IT support helps to accomplish another activity - the actual job - requires that we view operating a mobile IT support only as a secondary task. Thus, this secondary task has to be unobtrusive with respect to the primary task and must not exhaust the cognitive and physiological capabilities of the worker, such as attention for the device, available hands for the device operation, or just willingness to use the device while performing another activity.

Constraint patterns can help to identify the conditions of specific situations and thus describe these in an application-independent and domain-independent way. In focusing on constraint patterns - or sets of constraints - which affect the user interaction with the device and

Abstract

in mapping these constraints to usability information of user interfaces that were tested with these constraints, we can build up a generic description of the conditions of work situations that help to decide on the applicability of specific interfaces for certain situations.

Before computers were mobile, the interaction was mainly influenced by three components: the computing device, the user and the application that was supported by the computer. Now, we face two more categories that have to be added: the *environment* in which the device is used and the task that the device supports. Thus, the design of mobile IT support is limited by constraints with respect to the kind of *task* to be performed; the *application*, for which the task is performed; the influences caused by the *environment* on the execution of the task; the *device* chosen as the supporting hardware platform; and the abilities and work patterns of the *user*.

The implementation of the *Interaction Constraints Evaluation Tool (ICE-Tool)* based on the *Interaction Constraints Model* illustrates with real-world examples that matching work situations based on the constraints that impact these situations is a valid and workable approach. *ICE-Tool* demonstrates the concept and illustrates the necessary steps to identify work situations with similar work situations.

The actual benefits for the design process result from the possibility to match an identified constraint pattern to a set of constraints that occurred in a work situation of a previously conducted project and thus to retrieve usability information for the different user interface components in that application. In this way, it is possible to retrieve information about previous projects that did not appear similar to the current project, and were thus not considered as examples for the current design.

Acknowledgements

This research could not have been accomplished without the help of many people in my close and farther environment. At this point, I would like to thank some of them by mentioning, to all I could not mention here, please be assured that I always remember your efforts.

My greatest appreciation goes to Prof. James H. Garrett, Jr., my advisor for this dissertation. Somehow, Jim always found the right level of being advisor or colleague, as well as being a mixture between father figure, brother and good friend. I admire his professionalism and the way he manages all his jobs without sacrificing a life beyond.

I would like to thank my thesis committee: Profs. Burcu Akinci, Dan Siewiorek and Dr. Amin Hammad, who guided me through this research with valuable advice and suggestions.

I also thank Dr. Markus Klausner at Bosch, co-advisor of this work for more than two years, who brought in a more industry-driven attitude and helped me to keep realistic, real-world research goals. I would also like to thank Robert Bosch Corporation, Research and Technology Center North America, who financially supported this research for the last three years. My thanks also extend to the project partners at the German Bosch divisions: Dr.-Ing. Günter Nobis, Jürgen Anlauf and Oliver Welt.

A great help during my studies were the talks with my colleagues, fellow students and friends, especially Dr. Jirapon Sunkpho, who taught me how things work at CMU and in Pittsburgh, Jan Reinhardt for sharing his expertise in our research area, and of course Jacqueline Flor, my favorite office mate and friend. I also thank all the people that keep the department running more or less behind the scenes.

I am also thankful to all the people at Technical University of Darmstadt who prepared me very well in the area of Civil Engineering and especially in 'Bauinformatik' for the challenges I faced during this research.

My deepest gratefulness goes to Kirsten Kücherer, who took the adventure with me and supported me as much as she cheered me up and grew with me in these three years. Finally, I like to thank the two people who put the most effort in my education and development: my parents Peter and Renate Bürgy. This Ph.D. is also their success. Thank you!

Table of Contents

List of Figures	viii
List of Tables	xii
1. Introduction	1
<hr/>	
1.1. Problem Statement	1
1.2. Motivation	5
1.3. Objectives of this Research	6
1.4. Approach	7
1.5. Scope and Limitations	8
1.6. Contributions	9
1.7. Organization of Thesis	11
2. Mobile & Wearable CAE-Systems in Industrial Applications	12
<hr/>	
2.1. From Mobile to Portable to Wearable	12
2.2. Devices for Mobile Use	14
2.3. Speech-enabled m/w-CAE Systems	15
2.4. System Design and Systems Engineering of m/w-CAE Systems	16
2.5. Related Research on m/w-CAE Systems Development	19
3. Research Objectives and Contributions	24
<hr/>	
3.1. Objectives as Evaluation Criteria	24
3.2. Contributions	26
3.3. Research Path	28
An Interaction Constraints Model for m/w-CAE Systems in Industrial Applications	iv

4. The <i>Interaction Constraints Model</i>: Background	31
<hr/>	
4.1. Involved Research Areas	31
4.2. <i>Interaction Constraints Model</i>	33
4.3. Decision Support Systems	35
4.4. Constraints Definition	38
5. The <i>Interaction Constraints Model</i>: Description	40
<hr/>	
5.1. A Constraints Model	40
5.2. Task definition	42
5.3. Modeling Techniques	45
5.4. User Interface Modeling	47
5.5. Model Composition	50
5.6. <i>Constraints</i>	51
5.7. Work Location	60
5.8. Work Activity	61
5.9. Work Situation	62
5.10. User Interface	63
5.11. Use Cases in the <i>Interaction Constraints Model</i>	64
5.12. Evaluation of developed model	65
6. The <i>Interaction Constraints Model</i>: Implementation	68
<hr/>	
6.1. Implementation Requirements	68
6.2. Discussion of Implementation Techniques	69
6.3. Proof-of-Concept Implementation – <i>ICE-Tool</i>	76
6.4. Evaluation of <i>ICE-Tool</i>	80

7. Example Data from Real-World Projects	82
7.1. Description of Investigated Projects	82
7.2. SCWC – Vehicle Inspections	84
7.3. CISS-VR – End-of-Line Inspection in Manufacturing	89
7.4. ICMMS – Construction Progress Monitoring	92
7.5. MobileDCT – Landfill Monitoring	96
7.6. MIA – Bridge Inspections	100
7.7. Navigator – Aircraft Assembly	104
7.8. VuMan – Vehicle Inspection	107
7.9. Adtranz – Maintenance and Collaboration Tool	110
7.10. OSCAR – Crane Operator Assistance	113
7.11. Stars 3 – Power Plant Maintenance	116
7.12. Winspect – Preventive Maintenance	119
7.13. Digital Hard Hat – Multimedia Field Data	122
7.14. NOAH – Aid for Emergency Physicians	125
7.15. Deep Map – A Virtual Tourist Guide	128
8. Illustrative Usage Example	131
8.1. Identifying Work Situations	133
8.2. Definition of <i>Influences</i> and <i>Constraints</i>	141
8.3. Evaluation of <i>Constraints</i>	145
8.4. Entering / Retrieving User Feedback	149

9. The Interaction Constraints Model: Evaluation	151
9.1. Discussion of Proof-of-Concept Results in ICE-Tool	151
9.2. Initial Objectives and Anticipated Contributions	158
9.3. Discussion of Objectives	161
9.4. Contributions of this Research	164
9.5. Summary of Evaluation	166
10. Conclusions	167
10.1. Summary of Research	167
10.2. Outlook and Future Research Steps	169
Appendixes	175
Appendix A: Additional Reports for Illustrative Example	175
Appendix B: Constraint Patterns of ICE-Tool Examples	177
References	182

List of Figures

Figure 2.1: Derivatives of traditional user interfaces. From left to right: L3 Systems WristPC attachable keyboard [L3 Systems 2002], Handykey Twiddler [Handykey 2002], Finger Trackball [Extreme 2002].....	13
Figure 3.1: Approach for setting up the "system" (UML Activity Diagram)	29
Figure 4.1: Involved research fields for this research	32
Figure 4.2: DSS components after [Sage 91]	36
Figure 5.1: 4-level terminology shown in a Composite Activity Diagram	43
Figure 5.2: User Interaction Modeling derived from [Van Harmelen 2001] and changed to meet the needs for the <i>Interaction Constraints Model</i>	48
Figure 5.3: High-level composition of a <i>constraint / work situation model</i>	51
Figure 5.4: Composition of the <i>constraint object</i>	52
Figure 5.5: User Interface model.....	63
Figure 5.6: "Application" use cases that "include" the generic interaction use case "Identify object"	65
Figure 6.1: Constraints in a UML Class Diagram, constraints are added within brackets; OCL defines the notation of these constraints.	71
Figure 6.2: UML Activity Diagram using swimlanes. Each change of swimlanes describes either a user interface (#1-3) or an interface to another component of the IT infrastructure, here a database management system, DBMS (#4-6).....	72
Figure 6.3: ERD of the <i>Interaction Constraints Model</i> as implemented in <i>ICE-Tool</i>	77
Figure 6.4: Main Screen of the <i>Interaction Constraints Model</i> as implemented in <i>ICE-Tool</i>	78
Figure 6.5: An <i>ICE-Tool</i> report on previously implemented user interfaces.	79
Figure 7.1: Overview of projects divided by affiliation and research groups.....	83
Figure 7.2: Garage technicians with paper-based process (left) and using the SCWC 1 with the head-mounted display (middle, right)	85

List of Figures

Figure 7.3: (left to right) SCWC 1 graphical user interface, text-based LCD display of SCWC 2, hardware comparison of SCWC 1 (left in third image) and SCWC 2.....	85
Figure 7.4: (left to right) CISS-VR graphical user interface, Xybernaut MA IV, Fujitsu Stylistic	90
Figure 7.5: (from left to right) ICMMS graphical user interface, worker using the system, Xybernaut MA IV running ICMMS.....	93
Figure 7.6: (from left to right) MobileDCT graphical user interface, Xybernaut MA IV with Magellan GPS, system in use	97
Figure 7.7: (left to right) Bridge inspector in paper-based and wearable computer-supported process, CMU's TIA P wearable computer.....	101
Figure 7.8: (from left to right) Boeing's assembly shop, Navigator 2, system in use.....	105
Figure 7.9: (from left to right) VuMan graphical user interface and VuMan system in use.....	108
Figure 7.10: (from left to right) Adtranz graphical user interface, Adtranz system in use, and people mover train in tunnel	111
Figure 7.11: (from left to right) Oscar graphical user interface, crane with load, supply boat as target for the load.....	114
Figure 7.12: Stars 3 user interface prototypes for LCD display (left) and for the Augmented Reality system (right)	117
Figure 7.13: (from left to right) Winspect graphical user interface, Winspect sensor glove, and concept of handling large technical drawings.	120
Figure 7.14: (from left to right) Digital Hardhat graphical user interface, annotated image of construction site, and system in use.....	123
Figure 7.15: (from left to right) NOAH graphical user interface; NOAH in use on Xybernaut MA IV and on Motorola Forte	126
Figure 7.16: (from left to right) Deep Map graphical user interface, Deep Map system in use, and envisioned use on a PocketPC device.....	129
Figure 8.1: Use Case Diagram showing the two example use cases.....	132

List of Figures

Figure 8.2: Activity Diagram of Task “document with picture”; User Interfaces are indicated by swimlane changes #001-006, interfaces to the digital camera are required at #007-010.....	134
Figure 8.3: Activity Diagram of Task “document information about delivery”; User Interfaces are indicated at swimlane changes #101-108, interfaces to the digital camera are required at #109-111.....	135
Figure 8.4: Two different kinds of staircases: inside the construction (left) and leading to the outside of the construction (right).	137
Figure 8.5: Cherry picker workbasket with blueprints (left) and cherry picker with elevated basket (right).	138
Figure 8.6: <i>Work location</i> involving dust, water and concrete splashes.	139
Figure 8.7: Main Screen of the <i>ICE-Tool</i> as implemented with MS Access. <i>Work location</i> and <i>work activity</i> list boxes are located on the top of the screen, the <i>influence</i> , <i>constraint</i> and <i>user interface</i> lists at the bottom.....	140
Figure 8.8: Location input dialog	141
Figure 8.9: Activity input dialog	141
Figure 8.10: Influence input dialog	142
Figure 8.11: Influence input dialog with drop-down list of previously entered influences	142
Figure 8.12: <i>Task constraint</i> , related to <i>influence</i> “Poor Artificial Lighting” (caused by the closed staircase).	142
Figure 8.13: <i>Environment constraint</i> , related to <i>influence</i> “Poor Artificial Lighting” (caused by the closed staircase).	143
Figure 8.14: <i>User constraint</i> , related to <i>influence</i> “Poor Artificial Lighting” (caused by the closed staircase).	143
Figure 8.15: <i>Device constraint</i> , related to <i>influence</i> “Poor Artificial Lighting” (caused by the closed staircase).	144
Figure 8.16: <i>Application constraint</i> , related to <i>influence</i> “Form-based process” (caused by the nature of the incident report).....	144

List of Figures

Figure 8.17: <i>Application constraint</i> , related to <i>influence</i> “Form-based process” (caused by the nature of the concrete testing).....	146
Figure 8.18: <i>Constraints</i> for staircase <i>work situation</i> from ADL example.....	147
Figure 8.19: Reports selection dialog. The numbers indicate work situation with the same set of <i>constraints</i> in the specific <i>constraint category</i>	147
Figure 8.20: Report showing all work situations for a specific set of constraints.....	148
Figure 8.21: User interface input dialog	149
Figure 8.22: User interface implementation note dialog	149
Figure 8.23: Dialogs showing user interface information for the selected work situation (<i>filling in forms in the pit of a garage during the SCWC project</i>).....	150

List of Tables

Table 2.1: Hardware platforms and software development tools	17
Table 2.2: Design Time Comparison	17
Table 5.1: Task Categories in terms of relevance for interaction with mw-CAE Systems.....	44
Table 5.2: Attributes of <i>User Constraints</i>	54
Table 5.3: Attributes of <i>Environment Constraints</i>	55
Table 5.4: Attributes of <i>Task Constraints</i>	56
Table 5.5: Attributes of <i>Application Constraints</i>	58
Table 5.6: Attributes of <i>Device Constraints</i>	59
Table 6.1: Mapping of the considered implementation techniques to implementation requirements; from Section 6.1 (scale from “-“ over “o” to “+” for “not reached” over “sufficiently reached” to “completely reached”).....	74
Table 7.1: Location examples for SCWC project.....	86
Table 7.2: Activity examples for SCWC project	87
Table 7.4: Location examples for CISS-VR project	90
Table 7.5: Activity examples for CISS-VR project	91
Table 7.6: Implemented user interfaces for CISS-VR project.....	91
Table 7.7: Example locations of ICMMS project.....	94
Table 7.8: Example activities of ICMMS project	94
Table 7.9: Implemented user interfaces for ICMMS project	95
Table 7.10: Example locations of MobileDCT project.....	98
Table 7.11: Example activities of MobileDCT project	98
Table 7.12: Implemented user interfaces for MobileDCT project	99
Table 7.13: Example locations of MIA project	102
Table 7.14: Example activities of MIA project.....	102
Table 7.15: Implemented user interfaces for MIA project.....	103

List of Tables

Table 7.16: Example locations of Navigator project	105
Table 7.17: Example activities of Navigator project.....	106
Table 7.18: Implemented user interfaces for Navigator project.....	106
Table 7.19: Example locations of VuMan project	108
Table 7.20: Example activities of VuMan project.....	108
Table 7.21: Implemented user interfaces for VuMan project.....	109
Table 7.22: Example locations of Adtranz project	111
Table 7.23: Example activities of Adtranz project.....	112
Table 7.24: Implemented user interfaces for Adtranz project.....	112
Table 7.25: Example locations of OSCAR project.....	114
Table 7.26: Example activities of OSCAR project	114
Table 7.27: Implemented user interfaces for OSCAR project.....	115
Table 7.28: Example locations of Stars 3 project	117
Table 7.29: Example activities of Stars 3 project.....	118
Table 7.30: Implemented user interfaces for Stars 3 project.....	118
Table 7.31: Example locations of Winspect project.....	120
Table 7.32: Example activities of Winspect project	121
Table 7.33: Implemented user interfaces for Winspect project	121
Table 7.34: Example locations of Digital Hardhat project.....	123
Table 7.35: Example activities of Digital Hardhat project	123
Table 7.36: Implemented user interfaces for Digital Hardhat project	124
Table 7.37: Example locations of NOAH project	126
Table 7.38: Example activities of NOAH project.....	127
Table 7.39: Implemented user interfaces for NOAH project.....	127
Table 7.40: Example locations of Adtranz project	129
Table 7.41: Example activities of Adtranz project.....	130
Table 7.42: Implemented user interfaces for Adtranz project.....	130

List of Tables

Table 9.1: Anticipated results for tasks performed in office environments, handling text-based and table-based data.	152
Table 9.2: Anticipated results for tasks performed at an outside inspection location, during observation / inspection processes.....	153
Table 9.3: Matching <i>work situations</i> for tasks performed at a garage respectively a construction site, during a selection and a documentation process.	154
Table 9.4: Matching <i>work situations</i> for tasks performed at a moving vehicle and an apartment respectively, during a selection and a documentation process.....	154
Table 9.5: Matching <i>work situations</i> for tasks performed at a tunnel and on a highway at night respectively, during a communication process.	155
Table 9.6: Matching <i>work situations</i> for tasks performed at a manufacturing shop floor and in a restaurant respectively, during the use of online documentation.	155
Table 9.7: Matching <i>work situations</i> for tasks performed in a crane cabin and in a smoke-filled building respectively, during the use of computer-based guidance. ...	157
Table 9.8: Matching <i>work situations</i> for tasks performed at a landfill and in a distribution center respectively, during the use of computer-based guidance.	157
Table 9.9: Primary objectives for a system implementing the <i>Interaction Constraints Model</i> ..	158
Table 9.10: Secondary objectives for the <i>Interaction Constraints Model</i> itself.....	159
Table 9.11: Anticipated contributions of the <i>Interaction Constraints Model</i>	159
Table 9.12: Statements or assumptions made during the development of the <i>Interaction Constraints Model</i>	160

1. Introduction

" If we think of technology as a runaway monster, we can think of this as a way to tame the monster with a piece of itself. "

Steve Mann, University of Toronto, Canada, about wearable computers [Mann 2001].

The citation above implies two things: we cannot stop technology, the "runaway monster", but we can use wearable computers to close the gap between humans and technology. Bringing wearable computers to industrial applications as mobile IT support systems can be one example of this idea [Mann 2002]. Putting mobile and wearable computers for industrial applications into practice is the main motivation for this research. However, one of the keys for "taming the monster" is to improve the design of user interaction with mobile and wearable computers, so that they can truly support activities in industrial applications and help people who work in industrial environments to perform their tasks more efficiently and more effectively.

This chapter introduces the problem area and the motivation for this research on interaction with mobile and wearable computer-aided engineering systems (*m/w-CAE Systems*) in industrial applications and summarizes the objectives and the approach of this research. It concludes with an overview of the organization of this dissertation.

1.1. Problem Statement

Mobile and wearable computer-aided engineering systems can serve as Information Technology (IT) support systems for field-oriented tasks in industrial applications (referred to as 'mobile IT support'). These systems based on mobile or wearable computer systems can help to provide these workers with continuous and ubiquitous support for their information needs. In this way, these systems either support engineering tasks of field engineers or provide engineering

information to support mobile workers. The effective use of these systems depends highly on how the actual support is delivered to the worker. The interaction with mobile and wearable computers for industrial applications is influenced by two major factors: the users' demand for devices with a specific functionality, which helps them to perform their job faster and with higher quality; and the design of user interfaces that enable users to operate these devices while still performing their actual job. Both, the users' view, i.e., what kind of mobile IT support is needed, and the designers' view, i.e., which tools can help to design and implement *m/w-CAE Systems*, are important to understand the focus of this research.

1.1.1. User View: What Kind of IT Support is Needed?

Mobile IT support systems are ubiquitously used tools, which must perform in changing environments, and thus with changing requirements and constraints. These requirements and constraints range from physical requirements, such as size, weight, and dimensions of the device, to performance constraints, such as available bandwidth, data storage and power consumption, to usability constraints, such as those related to user interfaces and ergonomics.

The user's perception of mobile IT support is that it must help to perform a specific job and thus has to be a useful 'tool'. However, using this tool only is a secondary activity that helps to perform the primary activity, the user's purpose for being in the field. The use of mobile IT support must offer a benefit for the user over performing the job without the IT device or with other tools. First steps in this direction have been pen-based tablet computers, which have been used as a replacement for paper-based processes. These devices still require the user to carry around a clipboard-size 'tool', but make multiple data transitions between paper lists and stationary computers obsolete. The next step in mobile IT support is to enable interaction with mobile or wearable computers while actually performing the primary task. Most often in this situation, the user's hands are busy and the only way to interact with the device is via hands-free user interfaces, such as speech recognition. However, since these user interfaces are only

emerging, the design of these interfaces cannot yet be based on broad experience and thus it is difficult to apply the right support for a situation.

In the same way that a tool, such as a hammer is used to support specific work situations, such as driving a nail into a wall, we have to design mobile and wearable computers in a way that they can be used to sufficiently support a specific task. This implies that the interaction with this task-specific device has to be adapted to the situation in which it will be used. There are at least two approaches to address this need for task-specific device interaction: First, we could try to predict how and where the device will be used in this specific situation and design the user interfaces according to this scenario. Second, we could build intelligent, adaptive user interfaces that adapt autonomously to a given situation. To proceed with the tool analogy, we can identify three possible levels of support by the device:

- 1) The designer of the device predicts the usage situation and provides one specific interface for the interaction with the user, which is similar to a hammer or a wrench, which are interfaces that best support one specific task.

→ This is the currently taken approach for the design of mobile IT support.

- 2) The device provides a variety of interfaces from which the user can choose the best interface for a specific task. In the analogy, this would correspond to a Swiss army knife[®], which has different features, such as blades in different sizes, screwdrivers, and can openers.

→ This approach can be realized if we are able to identify all interfaces that are applicable for the tasks, which the user might face while using the device, so the user can choose from a variety of interfaces. This approach is an intermediate solution between the current approach and future intelligent, adaptive interfaces, and is that envisioned by this research.

- 3) The device provides a variety of interfaces from which the device automatically picks the best one to support the user in a specific situation. This works similar to self-adjusting pliers, which adjust to the object that is being grabbed as it is squeezed, both in terms of size and angle.

→ This is the approach at which intelligent, adaptive interfaces aim.

For all these cases, for the interaction design by humans as well as for intelligent, adaptive user interfaces, we need an underlying model that can describe task-specific situations and the applicability of a user interface component to different task-specific situations.

1.1.2. Developer View: Which Tools Support the Design?

Mobile IT support in industrial applications, i.e., the support of industrial processes by mobile or wearable computers, is only emerging. The design of these systems is not yet formalized, nor standardized. Furthermore, no standard software packages for mobile IT support exist, and most of the software applications for these devices are proprietary developments. Thus, each system design starts from scratch and requires immense efforts on requirements elicitation and field-testing.

Another problem that arises is how to support domain experts who participate in projects on mobile IT support and do not have sufficient background in systems engineering or software development. This is especially important since the relatively new mobility of computers has established new usage patterns in different new environments, where no computers have been used in the past. Interactions with the computer are now occurring under ever changing conditions imposed by the environment and the task that has to be supported: these interactions have to be designed in a new way, acknowledging the mobility of mobile computers. The decision on these user interface issues takes a significant amount of development time and no guidelines for the interaction design for mobile IT systems currently exist. Thus, it would help to have a description of the decision factors affecting interaction design for *m/w-CAE Systems* in industrial applications and a concept for mapping the given or anticipated constraints of specific situations to the applicability of existing and emerging user interaction mechanisms.

1.2. Motivation

There are three primary motivations for the research described in this dissertation: 1) the complexity of the interaction design for *m/w-CAE Systems* in industrial applications; 2) the lack of development and design tools for this interaction; and 3) the importance of a sound design process.

Interaction design for mobile IT support systems is a complex activity for the following reasons: Since many activities that are to be supported demand that the worker uses both hands, hands-free user interfaces have to be considered. Hands-free user interfaces are not yet well established, and with the exception of speech recognition technology, are not yet widely tested and approved. But even the use of speech technologies requires the incorporation of some new interaction design paradigms that go beyond the simple replacement of mouse clicks by speech commands. Also, the software applications now have to run on multiple computing platforms with different capabilities and user interaction means and have to be usable in different environments. Finally, mobile IT systems and their functionality are getting more complex and allow for the support of more complex and diverse tasks, which increases especially the time needed for the requirements analysis for such systems. All these aspects multiply the efforts that go into the design process of these devices, compared to stationary desktop applications, and motivates the need for systems that support design decisions for interaction with mobile IT support.

Some **development tools** exist that support the design of applications for mobile and wearable computers, such as frameworks for inspection applications [Sunkpho 2001], software development kits that allow for easier information synchronization with mobile devices [Wearix 2002] and even tools that allow for computer-aided user interface design for mobile support systems [Tangis 2002]. While all of these tools help in developing applications for mobile and wearable computers, and the last example even helps to facilitate the user interface design, there is still the need for supporting decisions concerning which interface to use in which situation. This decision is quite important because it may save developers from implementing unusable

interfaces in a mobile IT system and thus frustrating users at the start of a product development when irreparable damage to product reputation will be done.

The **importance of a sound design process** can best be expressed in terms of repair costs that are needed to fix design errors in different phases of a product development. Davis summarized studies of major companies [Davis 1993], which showed that finding and repairing an error in the requirements analysis phase is 200 times less expensive than at the time the product is already introduced. The costs of finding and repairing the error in unit or acceptance tests still exceeds the costs of finding these errors at design time by a factor of four to ten [Regnell 1999, p. 11], [Leffingwell 2000, p. 10]. This shows that we can improve the whole *product development process*, if we can improve the design process and reduce the error rate in this early phase. This is especially true if we can avoid a complete failure of a field test that was caused by an incorrect assumption on the usability of tested user interfaces. Although even failed field tests help to gain knowledge, we can save a lot of development time and thus can concentrate on other details of the design, if we avoid failures. If a field test should fail, the design team should at least have a means to document the issues learned in some kind of a knowledge base.

1.3. Objectives of this Research

The goal of this research is to **formally and generically describe** the applicability of **user interfaces with mobile and wearable computers for specific situations**; and thus to **improve the interaction design process** for *m/w-CAE Systems* and to **support system designers** in deciding which mode of interaction is appropriate for supporting a given activity in a given situation.

Specifically, I developed an interaction model and a technique of using this model that helps to speed up the design process by providing decision support about the right interaction mode with *m/w-CAE Systems* for a specific situation. This decision support is based on real-world

data and by considering the constraints that impact the given situation or usage scenario. Another result of this research is a formal and generic description of the applicability of user interfaces of mobile and wearable computers for specific situations, especially with respect to emerging interfaces and existing user interaction modes suitable for mobile and wearable computers, such as speech interaction. Chapter 3 will discuss these objectives in more detail.

1.4. Approach

The goal of this research was to develop an interaction design framework. This framework consists of an interaction model and a technique of identifying and reusing *constraint patterns* to decide on the usability of user interface components in specific situations. The approach is composed of the development of the interaction model, the description of the technique for identifying and reusing *constraint patterns*, and a proof-of-concept implementation that investigates the validity of the approach.

Interaction model. Developing the interaction model involved the investigation and the definition of the necessary components for the interaction design for mobile and wearable computers used as *m/w-CAE Systems* in industrial applications. In this activity, I identified and analyzed the *constraint categories* and the attributes that can describe specific *constraint categories* in a way that they can be clearly described for a specific situation. In contrast to a framework, which offers design reuse and code examples [Balzert 1996], [Johnson 1997], this model is not an implementation framework and can be reused and refined as a concept rather than a framework.

Technique of identifying and reusing *constraint patterns*. Besides the interaction model, I developed a technique that allows for identifying, describing, and reusing *constraint patterns* for specific situations in which *m/w-CAE Systems* will be used. This technique maps sets of *constraints* that describe specific situations to descriptions of user interface components and their usability in situations with similar sets of *constraints (constraint patterns)*.

Proof-of-concept implementation. After the development of the model and the technique, I implemented a proof-of-concept implementation that helped me to run demo queries on a set of previously entered project data. This data came from a study and evaluation of different mobile and wearable computer projects. In that way, I could verify the concept and the applicability of the model.

1.5. Scope and Limitations

The goal of this research was to develop a new approach for interaction design for *m/w-CAE Systems* that allows for making better-informed decisions about which user interface is applicable in which situations. This dissertation shows the complete approach, which is comprised of an interaction model and a technique for using this model, and ends with a proof-of-concept implementation that illustrates how this approach can be used for future decision support systems. This implementation is capable of demonstrating the main aspects of this research and to show the applicability of the interaction model in an actual implementation.

The terms 'approach' and 'concept' imply that the scope of this research is not to develop a complete decision support system. Instead, the goal is to develop and illustrate a formal process of identifying the different components that influence decisions about the right interface and to model these components in a way that they can be documented and reused. The scope of this research is limited to the investigation of the decision-making process on the applicability of user interfaces for *m/w-CAE Systems* in specific situations of industrial applications. The scope of this research does not include investigations of any user preferences for interfaces, which is based on the assumption that we can rule out interfaces that technically cannot function even before a user can choose or prefer a user interface. This gives the user the choice of all well-applicable interfaces in a specific situation.

The proof-of-concept implementation is limited by the fact that there is a limited number of real-world data available for the knowledge base or case base of the system. This results from

mobile and wearable computer systems being a relatively new research area and the lack of proper formalized documentation. Also, the intention of this proof-of-concept implementation is to verify the ideas of the model and to give advice for future implementations of the concepts rather than developing a complete system initially. Future implementations however, will need to include more cases, which can be collected and entered during the implementation or during the use of the system. These cases might be taken from additional systems described in the literature or from proprietary, company-own reports.

1.6. Contributions

The contributions of my research on the *Interaction Constraints Model* result from the developed theoretical model, the implementation of the proof-of-concept prototype, and the description of the approach or the technique of using the system itself, which can be adopted and transferred to other problem domains in the future.

The first contribution is the **decision support framework**. The *Interaction Constraints Model* can serve as the “model base” for a decision support system. Together with a case-base that contains information about previously conducted projects, the *Interaction Constraints Model* may be used to build a system that can aid system designers in deciding about the right user interface for the right situation. The above-mentioned model of user interaction will also serve as a **formal model of the requirements and constraints** for the description of user interaction with *m/w-CAE Systems*. Besides the model itself, I introduce a technique that helps in identifying different user interface needs during task analysis, assigning tasks to different task categories and eliciting requirements and constraints for the use of specific user interfaces.

The **generic description for categorizing tasks** contributes to the interaction design process, since no formal description could be found in the literature. Most of the design guidelines focus on traditional user interfaces, such as graphical user interfaces, and assume a computer that is used on a desktop or in a kiosk setup, which means the user and the computer system are

not moving. Furthermore, this research illustrates the transition of the interaction with the computer from a primary task (such as writing a business letter at a desktop computer) to a secondary task that supports another primary task (such as providing inspection information to a bridge inspector). This shows that the interaction paradigms and thus the design guidelines for this interaction have to be adapted.

The *Interaction Constraints Model* is my approach to give guidance in the design process so that future project teams will not have to start from scratch for the interaction design, and thus may be able to save one or two field-test iterations or to focus on different aspects in the field-test.

Another contribution is the collection of **real-world examples and user feedback** that I collected for the evaluation and validation of the model and its implementation. Since most projects that develop *m/w-CAE Systems* remain in the prototyping phase, it is essential to build up this case-base of previously made experience. This research contributes to this need in two ways: first, I describe the projects that went into the proof-of-concept in a general way, so as to give an overview of different approaches on using this technology; and second, I use specific data on these projects in the database of the implementation, which allows for comparing different implementations of user interaction for specific situations. In the future, the system will provide a means to collect more real-world data in an organized manner and to use this data as a case-base for supporting the design of future projects.

Finally, the system provides a formal description of the constraints that impact the interaction with a mobile or wearable device at operation time. This formal description can serve as a **basis for adaptable and adaptive interfaces** in the future. This is only one small step towards adaptive or intelligent user interfaces, since there are many influences affecting this decision. But knowing and being able to describe the constraints in a specific situation allows for automated decisions about the right interface for that specific situation. Describing these constraints for specific work situations (locations and activities) helps the system to be context-aware, and thus allow for better-informed decisions about the right user interaction. Of course, to

make such decisions, the user preferences, the system's performance, the connectivity and ergonomics of the system have to be investigated as well. These, however, might be integrated into the *Interaction Constraints Model* in the future and thus make it a more solid and complete model. For that purpose, the research in these areas has to be included in the *Interaction Constraints Model* to enhance and extend the data structure of the model.

1.7. Organization of Thesis

The remainder of this dissertation is organized as follows:

- Chapter 2 gives background information on mobile support systems in civil engineering and industrial applications and illustrates some of the issues in delivering mobile IT support to field-oriented tasks.
- Chapter 3 discusses in detail the objectives and anticipated contributions of this research and illustrates the research path I took.
- Chapters 4 and 5 give the requirements for, and the descriptions of, the *Interaction Constraints Model*, as I developed it in this research, which is followed by the description of the proof-of-concept implementation of the system in Chapter 6.
- Chapter 7 describes the projects from which I took the data to evaluate my model and Chapter 8 illustrates the usage of the model in a complete usage example.
- The last two chapters give an evaluation of the initial objectives and anticipated contributions (Chapter 9) and conclusions including an outlook on future research steps (Chapter 10).

2. Mobile & Wearable CAE-Systems in Industrial Applications

Field workers and field engineers in industrial environments do not usually have a desk at which to work and usually are in work environments that are more rugged and less clean than office environments. Currently, the common approach to support workers in industrial environments is to place centrally located computers near their actual workplace to enable the use of information technologies (IT) on demand. This information kiosk model does not conveniently and efficiently support industrial workers and engineers. These workers need to have access to the information they require for a task at the point of performance of that task [Billinghurst 1999, p.59]. Mobile and wearable computer-aided engineering systems (or *m/w-CAE Systems*) can help to provide these workers with a continuous and ubiquitous support for their information needs. These systems either support engineering tasks of field engineers or provide engineering information to support mobile workers [FieldWorker 2001]. *m/w-CAE Systems* are based on mobile or wearable computer systems and can thus also be referred to as mobile support systems or mobile IT support. These systems provide different forms of human-computer interaction (HCI) to sufficiently support field engineers and mobile workers during their job. Especially important are hands-free interaction modalities that allow the operation of a mobile or wearable computer without interrupting the primary task. Another issue is to determine how to provide information to the users of the system without distracting them from their actual job.

The following sections describe background information that I identified as essential for developing *m/w-CAE Systems* and the interaction with these devices, especially with respect to hands-free interaction with the computer, such as speech recognition.

2.1. From Mobile to Portable to Wearable

In recent years, computers have become smaller (from mainframes to personal computers) and eventually mobile (laptops). Some of these mobile computers are even portable in that they can be used during transport. The first laptops had to be placed on a solid surface

and not moved during usage, whereas today's machines can be carried and simultaneously operated. Since the operation of a standard laptop while walking or working is quite impractical and somehow dangerous, the wearable computer was invented [Thorp 1998]. Wearable computers are worn on a belt or carried in a pocket and have computing power that is equivalent to that of standard laptops. The important differences are the user interfaces that enable usage "on the move," such as head- and body-mounted displays, keyboards, and mouse devices or speech recognition and speech synthesis technologies. Figure 2.1 shows a few examples.

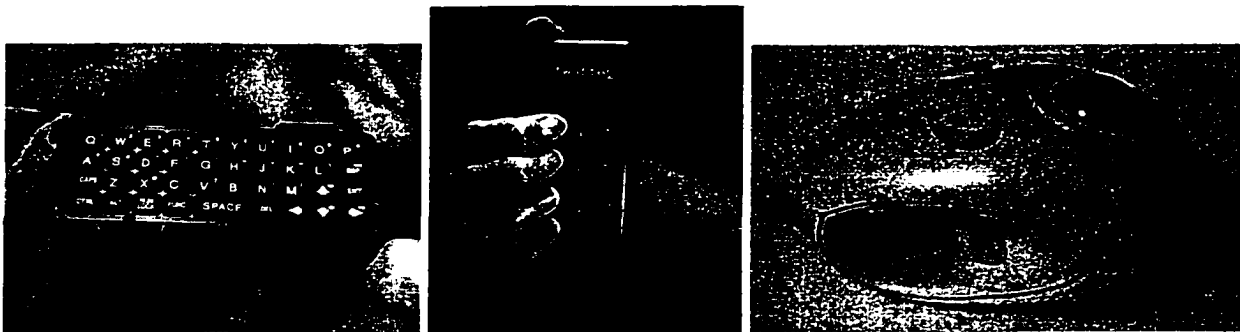


Figure 2.1: Derivatives of traditional user interfaces. From left to right: L3 Systems WristPC attachable keyboard [L3 Systems 2002], Handykey Twiddler [Handykey 2002], Finger Trackball [Extreme 2002]

There are two usage patterns for wearable computers. One is where the computer is used as an extension of the human. The device is always ready to respond to information retrieval and storage requests, and places address books, calendars, and other personal information within the user's view [Mann 1999]. The second usage pattern is that where the computer is used as a task-specific tool that helps workers to perform certain tasks of their job and is not always on but invoked when needed [Billinghurst 1999, p.60].

In contrast to wearable computers that evolved from powerful devices, another group of portable devices emerged from less powerful, smaller devices. Starting as organizers that could manage a handful of addresses and appointments, those handheld IT devices, now called personal digital assistants (PDA) and Pocket PCs, became more powerful and are currently

capable of running similar, or even the same, operating systems to those on sub-notebook computers. Thus, these devices are about to dominate the field of the portable computers with the advantage of being developed solely for mobile, portable use. However, due to current limitations in processing power, we cannot implement the same user interfaces as for wearable computers. Eventually, these devices might all merge together through a process referred to as “convergence” [Xybernaut 2001] In this convergence process, laptops, PDAs, pagers, and mobile phones merge into one mobile or wearable device. Thus, wearable computers should not only be seen as replacements for the laptop, but also as a means for communicating with colleagues, hotlines, customers, and keeping track of project management data, such as scheduling, address books, and knowledge bases.

2.2. Devices for Mobile Use

The great challenge for mobile IT devices is to design them for mobile use, i.e. for using them while walking or working. Many mobile input devices have been developed for use “on the move”, such as mobile, body-worn pointing devices or keyboards, scanners, or data gloves. But these interactions still involve using at least one hand. Some tasks, however, have to be performed using both hands, which makes the manual operation of an IT device distracting or even impossible. This is especially true for industrial applications, where the targeted users of mobile and wearable computers are workers conducting inspections, doing maintenance, or performing repair jobs [Najjar 1997], [Siegel 1997], [Ockerman 1998]. Most of the time, these people have to use both hands for their primary task, which is their actual job, and thus cannot use their hands to operate the device. Thus, hands-free input devices would be ideal for this human-computer interaction [Espisito 1997], [Van Dam 1997], [Billinghurst 1998], [Bass 2000].

One particular challenge in the design of usable mobile and wearable computer systems is to find the right interfaces to support users during specific tasks in specific environments [Buergy 2002]. Several research projects describe the need for guidelines that describe this development process [Calhoun 1998], [Baber 1999], but most of them take a high-level approach

to decide between traditional and several hands-free input modalities, such as speech control, lip motion recognition, eye tracking, gesture and face recognition [Cress 1997], and even the direct connection with the human brain [Chapin 1999]. However, with respect to industrial applications, speech recognition will likely remain the best usable hands-free interaction with computers in the foreseeable future. Even if such a technology emerges, the users of industrial applications will likely be slow to adopt it. Speech recognition technology has improved dramatically in the past several years, and in previous publications we have shown that command-based speech control could be implemented robustly enough for noisy industrial environments [Sunkpho 1998], [Buergy 2001], [Meissner 2001]. Thus, the most promising mobile support systems to date are speech-enabled devices, i.e. devices that offer speech as a hands-free interface when other interfaces are not an option.

2.3. Speech-enabled m/w-CAE Systems

Speech technology consists of two categories: speech recognition and speech synthesis. Speech recognition enables the computer to understand a human's voice, and thus to be controlled by that human without a keyboard or pointing device [Kamm 1994], [Nasbaum 1995]. Speech synthesis offers contextual feedback to the user without the use of a display. Hence, speech recognition and synthesis enable "hands-free" and "eyes-free" operation of IT devices, respectively [Damper 1993], [Cohen 1994], [Denecke 1997], [Rosenfeld 2000]. Probably the most unobtrusive use of a mobile or wearable device is a solely audio-based device, that only uses speech recognition and speech synthesis as user interfaces [Vocollect 2001]. However, in many industrial applications the ambient noise does not allow the use of speech recognition and in many applications graphical data is required and thus some kind of graphical feedback should be incorporated in the user interface device [Buergy 2001a].

In industrial environments, hands-free operation of IT devices is sometimes essential to continue the primary task, the actual work activity to be performed. In order to not distract the

worker from his or her job, or worse put them in danger, hands-free access to data and eyes-free feedback from the system are important aspects of mobile support systems.

Industrial environments are not speech recognition-friendly environments. It takes some effort to *identify the right level of speech interaction* and to optimize speech control to achieve acceptable recognition performance. This can be done either by optimizing the speech engine itself, or on the systems engineering side, by intelligently applying speech technologies and mixing speech interfaces with traditional user interfaces based on the conditions of the situation at hand.

2.4. System Design and Systems Engineering of m/w-CAE Systems

One of the drivers for this research has been the experience I gained during the development of several task-specific wearable computers for industrial applications. The speech interaction design took a major portion of the design time. This section describes the challenges that including speech interfaces brings to a wearable computer development project; it demonstrates the importance of knowing which interface can work and thus which interface can be successfully implemented for a specific work situation that requires mobile IT support. In knowing the development efforts for user interaction design, one can quickly see the need for an aid that facilitates and speeds up the interaction design process for *m/w-CAE Systems*.

Our research group in the m/w-CAE Systems Lab [m/w-CAE Systems 2002] has conducted several research projects on speech-controlled wearable computer systems for industrial applications. The projects addressed different domains (automotive, manufacturing, and civil engineering) and targeted different applications. Table 2.1 shows the hardware platforms and the software development tools used for these projects. CISS-VR is an end-of-line support system using speech-controlled virtual reality 3D objects; ICMMS supports progress monitoring on construction sites [Reinhardt 2000]; mDCT is a support system for mobile landfill monitoring [Meissner 2001], MIA is a system for speech-controlled generation of bridge inspection reports

[Sunkpho 1998]; and SCWC is a speech-controlled mobile support system for vehicle safety inspections [Buergy 2001]. In these projects, we talked to the envisioned users of the systems and identified speech control as the most applicable currently available hands-free interaction modality to enhance traditional user interfaces.

	CISS-VR	ICMMS	mDCT	MIA	SCWC
Wearable Hardware	Xybr. MA IV	Xybr. MA IV	Xybr. MA IV, Magellan GPS	Via II PC	Xybr. MA IV Proprietary HW
Development Language	Java	Java	Visual Basic	Visual Basic	Visual Basic
Speech Engine	L&H ASR 1600	IBM ViaVoice	Dragon Naturally Speaking	Dragon Nat. Speaking IBM ViaVoice	L&H ASR 1600 L&H ASR 1500

Table 2.1: Hardware platforms and software development tools

We were able to compare the development efforts for the different components of the wearable computer systems. Table 2.2 shows the comparison of the development times needed for the different design and implementation tasks of each project and the actual share of development time spent for each part of the system, such as hardware architecture design and implementation, the actual software application, the graphical user interface (GUI), and the speech interaction development.

Design & Development Time	CISS-VR	ICMMS	mDCT	MIA	SCWC
Hardware Architecture	20%	10%	25%	25%	40 %
Application Logic	30%	35%	40%	30%	20 %
Graphical User Interface	25%	30%	15%	15%	10 %
Speech Interaction	25%	25%	20%	30%	30 %

Table 2.2: Design Time Comparison

From this comparison, the need for making the interaction design process, and in particular the speech interaction design, faster and more efficient is obvious: the (speech) interaction design takes a major share of the development time of the system, but is to a large extent application-independent and thus could be formalized and supported similarly to existing GUI design formalizations and guidelines.

The numbers in Table 2.2 show that the design of the speech interaction took at least as much time as the GUI design (with the exception of the ICMMS, which included a complex GUI with interactive construction drawings). *Speech interaction design also took nearly as much time as the hardware architecture design (either proprietary design or systems design with COTS components), which shows the potential in reducing development times with a formalized process for speech interaction design for mobile systems.*

There are guidelines on how to build speech dialogs [Markowitz 1996], [Bernsen 1997], but most of them were set up for an office or lab environment where there is low ambient noise, the user sits in a chair and has his or her full attention devoted to the application (mostly using word processors, spreadsheets, or browsing the web).

Possible solutions for supporting the speech interaction design process would either be a decision support system that suggests forms of interaction to the developer or even an automatic generation tool for interaction design. Such systems can take as input the specifics about the user, *the device being used, the application being performed, the environment in which the application is being used, and the tasks to be performed, and either suggest solution strategies or create application frameworks that have to be implemented and optimized.* Similar systems already exist or are in progress for GUI development [Shirogane 1998], [Bredenfeld 2000]. However, the first step for such a tool has to be a formalized description of these input parameters, especially with respect to the performed tasks, and a common model of user interaction in this area. Based on this model, a decision support system can be developed that might in the future evolve to a more automated design tool.

2.5. Related Research on m/w-CAE Systems Development

Chapter 7 presents a more detailed description of research efforts related to *m/w-CAE Systems* development in which 14 projects on mobile and wearable computer systems developed by different research groups are discussed. This section focuses more on the research topics of the area of mobile and wearable computing in industrial applications than on actual examples. The design and development of *m/w-CAE Systems* involves several different research topics that have different foci depending on the kind of activity that will be supported and the environment in which this activity is performed: hardware; user interfaces, context-awareness, and Augmented Reality (AR). The following subsections discuss each of these main research topics for *m/w-CAE Systems* development.

2.5.1. Hardware

A lot of research is done concerning the development of new mobile and wearable computer systems. While most of the laptop computer or PDA manufacturers and their products are well known to the public, such as IBM, Dell, Compaq, Hewlett-Packard, and Palm [IBM 2002], [Dell 2002], [Compaq 2002], [Hewlett-Packard 2002], [Palm 2002], many other vendors are trying to reach niche markets in developing pen tablet computers, such as Fujitsu [Fujitsu 2002], or Walkabout [Walkabout 2002], or converged mobile phone and PDAs, such as PC-EPhone [PC-EPhone 2002], or Handspring [Handspring 2002]. The commercial wearable computer market is still quite narrow with mainly Xybernaut [Xybernaut 2002] and ViA [ViA 2002] offering acceptable products for the use in industrial applications. In the academic research arena, there are a few groups working on improved wearable computer systems: the WearableGroup at Carnegie Mellon University [WearableGroup 2002], the MIT Wearable Computing at the MIT Media Lab at the Massachusetts Institute of Technology [MIT Wearables 2002], the Georgia Tech wearable group [GT Wearables], and Steve Mann's group at the University of Toronto [Mann 2001a].

Although the research efforts of the different groups are very diverse, they face the same problems in developing hardware systems for mobile and wearable use. The following list is an overview of the main issues.

Size and weight: Size and weight are important if a user has to carry the device for a long time, has to carry it in areas that are difficult to access, or just has to protect it from being damaged by environmental influences. The compromise here is the functionality of the device versus the physical dimensions. [Buergy 2000, pp. 64-71]

Storage and Memory: If data has to be stored locally on the mobile unit, it is important to have enough storage capacity. An alternative is to access data from a server by means of networking capabilities [Smailagic 2000].

Networking Capabilities: For a connection to a server or other mobile units, the network bandwidth of the connection to a wireless network has to be determined. If a synchronization of the remotely collected data is only necessary in the office, one can still use mobile or wearable computers with no wireless networking capabilities. [Brewer 1998]

Processing Power / RAM: Depending on what kind of applications should run on the mobile unit and what software has to be executed, the processing power has to be chosen. For example running a speech engine requires powerful processors and some amount of Random Access Memory (RAM). If we go beyond that and consider Augmented Reality as a desired functionality of the system, we might currently not be able to run it on a device with an acceptable size. [Piekarski 2001]

Power Capacity: Since mobile devices run on battery, the battery's capacity will set the recharging cycle and thus cause interruptions in the workflow for recharging or changing batteries. Thus, the right balance between processing power, performance and power consumption plays a great role in the acceptance of the device [Smailagic 2001].

Input / output modalities: For the human-computer interaction, it is very important to choose the right input / output interfaces to support users appropriately. For example, the existence and the size of a display and the connectivity to other peripherals affects both cost and usability [Bass 1997]. This last issue is closely connected to the next section, which describes the issues of the counterparts of hardware input and output interfaces, the user interfaces in the sense of software interfaces.

2.5.2. Multi-Modal User Interfaces

User interfaces are a topic that comprises a lot of different research areas. Besides the hardware interfaces, it is important to design the software in a manner that truly supports the user of the device in reaching these desired goals for interaction [Myers 1998]. Therefore, human-computer interaction [Johnson 1992] involves many disciplines from workflow analysis and management, psychology, social sciences, graphic design, systems engineering and system design and software engineering. The combinations of different interaction modalities, such as keyboard, pointing device, speech recognition, lip reading, or handwriting recognition, are called multimodal user interfaces [Vo 1998, p.5]. Multi-modal user interaction can be sequential, uncoordinated simultaneous, or coordinated simultaneous [Niklfeld 2001, p.3]. Sequential multimodal input means that the user can only use different input modalities one after another for a specific step; uncoordinated simultaneous input means that the user can choose at any time which modality to use, but only one modality will result in an action; and coordinated simultaneous input means that the actions triggered by the different modalities will be executed according to their timestamps and importance. The challenge for the system designer is to find the right combination of different modalities to allow the user to choose from the most suitable set of modalities to perform an action. In applications for *m/w-CAE Systems* the conditions for the user interaction vary with the changing usage locations. Each of the different modalities performs differently with these conditions. Hence, the designer of multimodal user interaction with mobile

and wearable computers has to consider different modalities as well as different usage conditions.

2.5.3. Context-Awareness

Context-awareness of *m/w-CAE Systems* is another essential research area that will facilitate and enhance the use of mobile IT systems. Context-awareness helps to provide information to the user that relates to the location of the user and the task that the user is currently executing [Stamer 1998]. Therefore, the system has to get information about its location by means, such as Global Positioning Systems (GPS), video cameras, or wirelessly connected orientation beacons. If the task the user executes is not related to the device and thus not recognizable by means of logging the interaction behavior of the user, other possibilities, such as monitoring the physical conditions, such as heart rate or body temperature, have to be considered. Context-aware devices can retrieve and process information, such as information about tourist attractions that a tourist with an electronic tourist guide is passing [Malaka 1999]. The device can for example show the way to the nearest restaurant if the tourist indicates to need a rest. If the environment around the user of a context-aware device is smart, it can adapt itself to the needs of its inhabitants and provide specific services to them [Anind 1999]. Context-awareness is a key requirement for Augmented Reality, which will provide the most helpful and unobtrusive way to deliver mobile IT support to the user.

2.5.4. Augmented Reality

Augmented Reality (AR) combines the concepts of Virtual Reality (VR) and context-awareness, in mapping a view of a virtual world to the real world based on the context of the user. While in VR, a user can move in a complete artificial world, AR only incorporates specific objects and includes them in the view of the user [Azuma 1997, p.2]. Typically AR systems use some kind of a head-worn display to overlay the user's view with the virtual elements, either with an optical see-through display, which allows seeing the real world directly around the virtual objects, or with a video see-through display, which shows recorded image of the real world in which the

objects are placed [Azuma 2001, p.2]. Optical see-through shows the virtual objects of the augmentation “hanging” behind the real-world view, due to the latency of the registering and image processing of the system. In the video-see through the complete image shows this kind of latency, which makes it harder to interact with moving real-world objects. Together with hands-free user interfaces, AR will be a real enabler for applications for mobile and wearable computers, especially in industrial environments. In industrial applications information, which is related to the environment is needed, most of the time but at the same time, the view of the user of a mobile IT device must not be obstructed completely.

Possible applications for AR and wearable computers, reach from manufacturing and medical applications to inspections and quality assurance [Barfield 2001]. In construction, AR can be used to augment infrastructure components, such as information of buildings material and health information of bridge girders [Hammad 2002]. The advantage of more broad-scale augmentation is that it can be done with the currently available accuracy of the registering, which is dependent on accuracy GPS data, currently around 3 meters. However, if AR systems run in defined application areas, differential GPS, which compares to a secondary local sender, can help to increase the accuracy to an acceptable range (to the centimeter range). The main drawback of GPS systems results from the low power of the satellite signals, which does not allow for a use indoors and even restricts the use between high-rise buildings or in deep valleys.

3. Research Objectives and Contributions

In the following list of objectives, I will call the outcome of my research a “system”, which is a rather broad expression and can be applied to nearly anything that is composed of some subparts or “sub-systems” respectively. The terminology will be clearer and more precise as I proceed in describing the development process and the various components that influence the design of the “system”.

3.1. Objectives as Evaluation Criteria

This chapter will describe the primary and secondary objectives (enumerated as “O-xx”) for the “system” that map the envisioned functionality with some of the usage mechanisms of the system. Based on these initial objectives, I will further explain the contributions of the research and describe the research path taken.

3.1.1. Primary Objectives

Primary objectives are the basic objectives that the system, which is developed in this research, should reach:

O-01: The model and system should formally and generically describe the applicability of user interfaces with mobile and wearable computers for specific situations.

→ This is the main objective of this research. Setting up a model and a concept for a formal description of usage scenarios for mobile and wearable computers contributes in improving the decision making process during the system design.

→ Since no formalization has been approached to compare different usage scenarios, existing design knowledge of previous projects is hard to access and compare.

→ Such a description will help to automate the decision making process (see *O-03*), which will allow the development of adaptive interfaces in the future.

*O-02: The use of the system developed in this research should **support system designers** in deciding which mode of interaction is appropriate for a given activity in a given situation.*

→ Software engineers often do not have the insight into the specific domain to fully understand the different situations, and thus need a guideline in deciding when to use which interaction mode.

→ Domain experts involved in the project, on the other hand, might not be able to perform state-of-the-art requirements engineering tasks because of a lack of time, resources, and knowledge.

*O-03: Using the system should **speed up the design process** by providing real-world data, use cases or patterns on how to implement given interaction modes, such as speech technologies.*

→ Until now, the design for speech-enabled software applications has been performed from scratch each time. A guideline for this process will allow for a more efficient design process;

Primary objectives will be verified and evaluated in the case study in Chapters 7-9

3.1.2. Secondary Objectives

Secondary objectives are the demands on the (data) model behind the envisioned system to enable the primary objectives:

*O-04: The system has to be **domain-independent** (domain-neutral).*

→ User interfaces should always be domain-independent, since the presentation layer should be separated from the actual application logic and the domain model [Fowler 1997, p.247], which reflects the basic idea of the Model-View-Controller (MVC) software architecture; thus, the envisioned system or the use of this system has to be domain-independent, too.

*O-05: The system has to be **implementation-independent**.*

→ Software development strategies vary between different development teams. Since there is a great variety of implementation tools, such as programming languages, platforms and paradigms, the system should not restrict to any one of these strategies.

*O-06: The system has to be **applicable to multi-modal interfaces**.*

→ For this research the scope is set to speech technologies as the focus. This is because speech technologies, at the moment, are the most advanced but yet not very broadly implemented hands-free user interaction mode. In the future however, new (hands-free) user interfaces will be developed and thus, the system should be applicable for other modalities in the future.

*O-07: The underlying data model has to be **generic** enough to fit a significant set of collected data (requirements or constraints), but **narrow** enough so as not to be too fuzzy.*

→ As stated in [Gudgeirsson 2000, p.10] a description language cannot cover all possible circumstances, but it should cover a majority of possible cases.

*O-08: The underlying data model has to be **understandable by domain experts** but also **machine-readable** for future use in adaptive interfaces.*

→ This objective is derived from objectives *O-01* and *O-03*.

Secondary objectives will be verified in the evaluation of the model definition and model representation provided in Chapters 5 and 6.

3.2. Contributions

The following is a list of contributions (enumerated as “*C-xx*”) this research has made and that will be evaluated and further discussed in chapter 9:

*C-01. The main contribution of this research is a **decision support framework** that supports domain experts with little development experience (i.e. non-CS developers) in realizing mobile and wearable computer systems for their domain.*

→ This contribution builds up on C-02 – C-04, which themselves are a basis for C-05; on the other hand, this contribution also helps software engineers that are not familiar with the target domain to gain an insight on the activities of the user much more efficiently.

→ Knowing the problems of the target domain can be more important than the actual implementation skills of the developer. Furthermore, sufficient requirements management and correct software specifications are crucial for the success of a project [Leffingwell 2000, p.7-13]

*C-02. A **formal model of the requirements and constraints** of the different design factors for the design of user interaction (esp. speech interaction) with mobile and wearable computers is developed. This description facilitates the interaction design process with categorized and formalized requirements and constraints.*

→ Only on the basis of formal descriptions it is possible to unambiguously describe system specification and analysis model of a specific design. [Bruegge 2000, p.99].

*C-03. As “tasks” are one of the needed design factors, this research provides a **generic description for categorizing tasks** and their relationships to other design factors (device, environment, etc.) and a guideline on how to map these tasks to the formal description mentioned in contribution C-02.*

→ This contribution helps in performing task analyses. [Johnson 1992]

*C-04. This research provides a case-base of **real-world examples and user feedback** that has been collected during the development of the proof-of-concept implementation.*

→ These real-world experiences will act as template and “good and bad practices”.

*C-05. As mentioned in C-01, this generic model can be used as a **basis for adaptable and adaptive interfaces** in the future.*

→ As users have different preferences and goals [Thompson 2000, p.2], and additional constraints caused by the environment influence the user interaction, more intelligent interfaces would definitely increase the usability of the systems.

3.3. Research Path

The idea of this research matured during real-world industry projects, e.g. in [Buegy 2001], that showed the necessity of the envisioned system. Based on these experiences and the literature research that I conducted, I first developed a general knowledge on what is needed. The development of this “system” comprised three major iteratively performed steps: the definition of the requirements for building the “system”, the definition of the “system’s” theoretical model, and the evaluation of the system against the requirements and objectives. These steps are described in more detail in the following three subsections.

3.3.1. Define the requirements for building the “system”

The first step was to gather the objectives the system should fulfill to meet the anticipated functionality. These objectives are listed in Chapter 3.1 as the primary objectives *O-01 - O-03* and the secondary objectives *O-04 – O-08*. These objectives implied building a system that somehow supports software developers in their practical work.

3.3.2. Define the “system’s” theoretical model

To define such a system, an underlying theoretical model is needed that describes the scope and capabilities of the system and how the information it contains is organized. The development of the system is described in Chapters 5 and 6.

3.3.3. Evaluate the “system” against the requirements and objectives

To test the system on its applicability, the system has to be implemented in a prototyped solution and evaluated against the requirements and the objectives. Chapters 5 and 6 give the individual comparison against the objectives, and Chapters 8 and 9 provides an idea of the quality of this research in showing some real-world evaluation.

Figure 3.1 shows a UML Activity Diagram of the single steps that I performed during this development process.

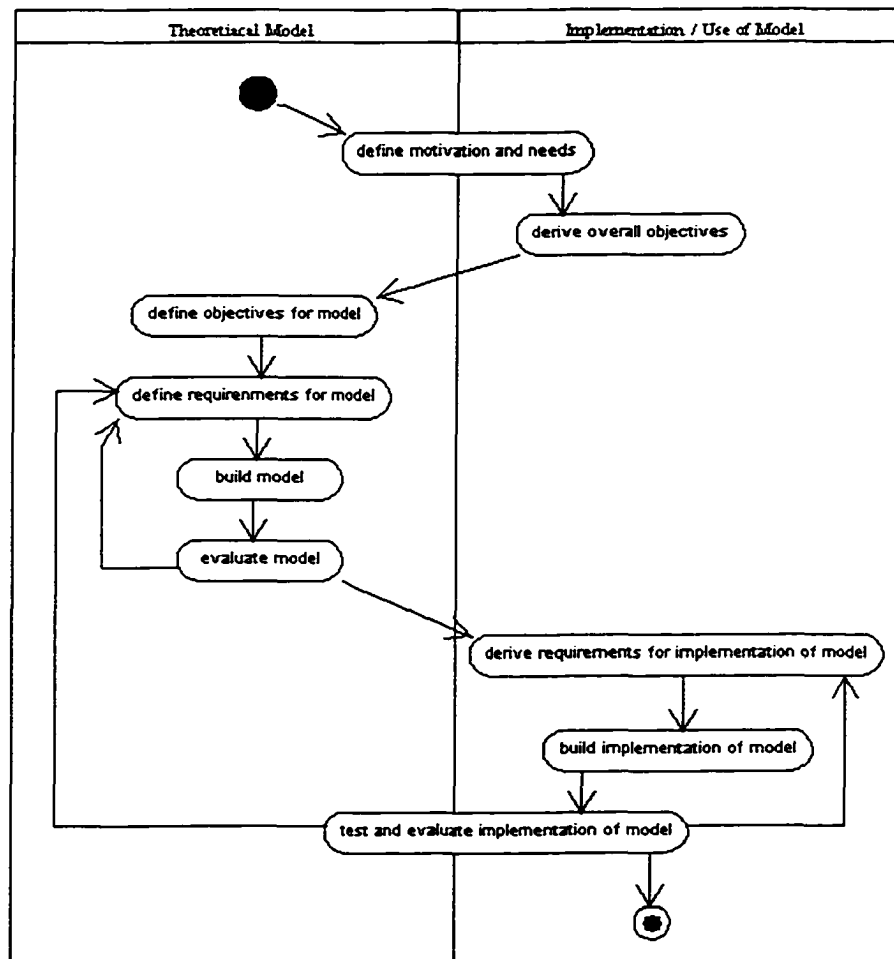


Figure 3.1: Approach for setting up the "system" (UML Activity Diagram)

Chapter 3: Research Objectives and Contributions

In setting up objectives and requirements first and then evaluating the developed model and system, I ensured that the quality of the developed “system” reached the envisioned contributions. Furthermore, the design decisions and the corresponding advantages and disadvantages of the design can clearly be followed throughout the thesis.

4. The *Interaction Constraints Model*: Background

Derived from the Objectives *O-01 – O-08* (see Chapter 3.1), it is possible to define requirements that the formal interaction model has to fulfill as to meet these objectives. During this process, and during the literature research that enabled this research, I found a lot of facts, which are related to this work. Since these facts are my own conclusions, but are basis for the requirements elicitation of the system, I will list them in each section as “Statements” (enumerated as “S-xx”).

4.1. Involved Research Areas

First, I would like to list the different research areas and fields that have to be considered for defining a usable speech interaction model. These fields are subcategories of computer science and engineering and reflect the areas of which I drew most of the underlying modeling and reasoning techniques. Figure 4.1 visualizes this dependency.

- Human-Computer Interaction (HCI) with Workflow Management (WfM), Task Analysis, and user interface design;
 - ➔ In HCI, there has been a lot of research towards non-mobile environments, but not yet a lot of research on interaction with mobile and wearable computers. However, we can use existing concepts and extend these to meet the mobility aspects of mobile and wearable computers.
- Requirements Engineering (RE) with requirements elicitation, requirements management;
 - ➔ RE focuses mainly on design time and not on operation time. Especially at operation time, we see a lot of “Constraints” that influence the interaction between users and the computing systems.

- Systems Engineering and System Design with design guidelines, rapid prototyping, design re-use.
 - ➔ The design of mobile and wearable computers should follow systems engineering concepts to reflect the integration of hardware, software, and the application (the activities, the system is intended to support).
- Software Engineering with Software architectures and user interaction concepts (MVC, Client/Server, etc.).
 - ➔ Software Engineering offers a lot of concepts, paradigms and tools, which facilitate the formal description and design of software that can be transferred to and used by other disciplines as well.

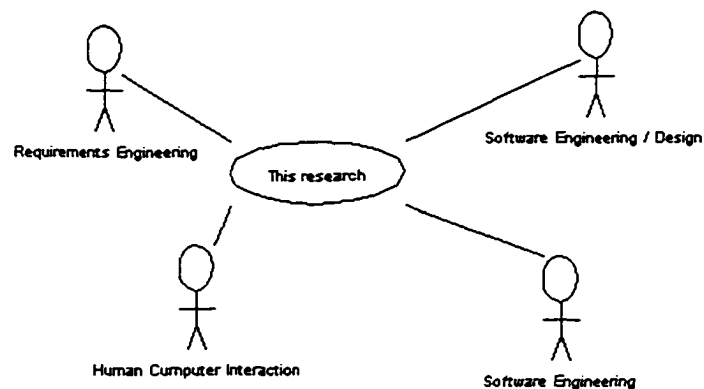


Figure 4.1: Involved research fields for this research

S-01: The design of interaction between users and mobile and wearable computers involves several disciplines, and thus is an activity that requires a huge amount of training.

- ➔ If domain experts rather than computer scientists should do this design, a predefined interaction model can help with providing some templates or guidance for this process.

4.2. Interaction Constraints Model

In Section 3.3, I stated the need for a theoretical model that is the basis for the “system”. For a better understanding of the kind of model that I developed, this chapter provides a description of what I call the “Interaction Constraints Model”.

Interaction between two agents usually follows rules and patterns, which is especially true for human-computer interaction (HCI), because the actions of a computer (or a computer interface) are limited. This limitation is caused by the capabilities of the machine itself and the software that is running on the machine. Thus, a computer can only execute actions that are defined and enabled by the software, and the human-computer interaction is limited to these defined actions. Furthermore, the model within the software that defines and enables the actions restricts the interaction and can be referred to as the “interaction model”. Beaudouin-Lafon defines interaction models as follows: “An interaction model is a set of principles, rules and properties that guide the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the ‘look and feel’ of the interaction from the user’s perspective. Properties of the interaction model can be used to evaluate specific interaction designs” [Beaudouin-Lafon 2000, p.446].

Through mobile and wearable computing, this interaction model has changed compared to the former stationary use of computers – mainly in office environments. Now that the interaction with a computer moved away from the desk in an office, we also have to consider other actions that are not directly interaction with the machine. Since dealing with the computer becomes a secondary task, “Direct Manipulation” becomes even more desirable. Users will not accept “the distractions of dealing with tedious interface concepts” [Shneiderman 1997, p.3]. The idea is that not moving a mouse or typing on a keyboard is the actual intension of the user, but to draw a blueprint or write a business letter. Through mobile and wearable computing this idea becomes more obvious: not carrying around a computer and caring about the ways to interact with it is the intension of the user, but to get support for an inspection process or instructions for a

complicated installation. Thus, we have to add the activities that are not considered interaction with the machine, but still may influence this interaction.

S-02: Even actions that are not considered interaction with the machine have to be added to the interaction model for interaction with mobile and wearable computers to reflect the fact that operating these computers is only a supplementary task and not the primary goal of the user.

→ Only if all actions that are conducted while carrying or wearing the computer are considered, all influences resulting from them can be identified.

The purpose of the model is to categorize and describe interaction activities and their relationships; i.e. if we can document several situations that involve typical sets of activities, software designers (or the machine) can later use them as templates. This way, the model can provide patterns on how to design the interaction with the machine, and thus build the basis for the support system for developing software for mobile and wearable computers. Also, a clear model of the different activities conducted by the user, will help to develop adaptive user interfaces that react on the user's activities and the environment in the future. This of course will need a huge amount of (artificial) intelligence on the machine.

So far, most of the software applications only offered interaction with the machine through traditional interfaces, sometimes called WIMP interfaces, where "WIMP" refers to Windows, Icons, Mouse, and Point-and-click [Van Dam 1997, p.63]. Wearable computing though, will most often demand a "hands-free" interface, i.e. an interface that the user controls without hands. By now, speech technology (speech recognition and speech synthesis) – in my opinion - is the most advanced hands-free user interface, and thus the one that most likely will be applicable to wearable computers in industrial applications, at least in the foreseeable future. This is the reason why I set the focus of the interaction model on speech. However, an "Interaction Constraints Model" that focuses on the implementation of speech technologies, already covers multi-modal interfaces since it incorporates traditional (WIMP) interfaces. Furthermore, objective O-06 asks for an interaction model that is extendable to additional interfaces in the future.

S-03: The design of speech interaction can serve as one example for a multi-modal user interface design for use in mobile and wearable computing.

→ If we can model speech and the system is extendable, we can include additional modalities in the future.

4.3. Decision Support Systems

Decision support systems (DSS) are intended to help users making decisions about specific problems. Underlying these systems are models of the problem area and analytical and scientific methods by which to approach the problem. For this process, users and other applications provide relevant input parameters. Such DSS “can execute, interpret, visualize, and interactively analyze these models over multiple scenarios” [Bhargava 1999, p.31]. According to [Druzdzal 2000, p.7] who refers to [Sage 1991], a decision support system is comprised of three parts:

- A Database Management System (DBMS);
- A Model-Base Management System (MBMS); and
- A Dialog Generation and Management System (DGMS).

The DBMS contains a large amount of data that will be analyzed according to a model chosen from the MBMS. Models contained here are independent from each other and provide different views of the data contained in the DBMS. The DGMS contains the user interfaces for the decision support system and allows users to create their own models and dialogs to analyze the database. Figure 4.2 shows the relationships between the three DSS components.

In my research, I tested four of the commercial DSS development systems (“Analytica”, “Decide Right for Windows”, “Decision Pro”, and “Logical Decisions for Windows”) mentioned in [Bhargava 1999, p.33], but I soon found out that decision support for user interaction is hard to fit into these commercial systems that are mostly based on logical or statistical models. Since not too much experience can be found on speech interaction in literature, it is quite difficult to apply

heuristic and empirical values to some of the parameters. For example: the question of how much ambient noise would be acceptable to still run a speech recognition engine depends on many other factors, such as the microphone that is used, the speech engine, and the preferences of the user. It would take a lot of knowledge in DSS design and a lot of analysis time to add these parameters in such a commercial system. This is even truer for domain experts that seek support for just a few problems in their software design. Furthermore, the decision of whether to use or not to use speech (or any other interaction means) in a certain situation is a binary decision. Thus, if the answer can only be “yes” or “no”, in my opinion, it makes more sense to model the relationships between the input parameters than assigning priorities or values to them.

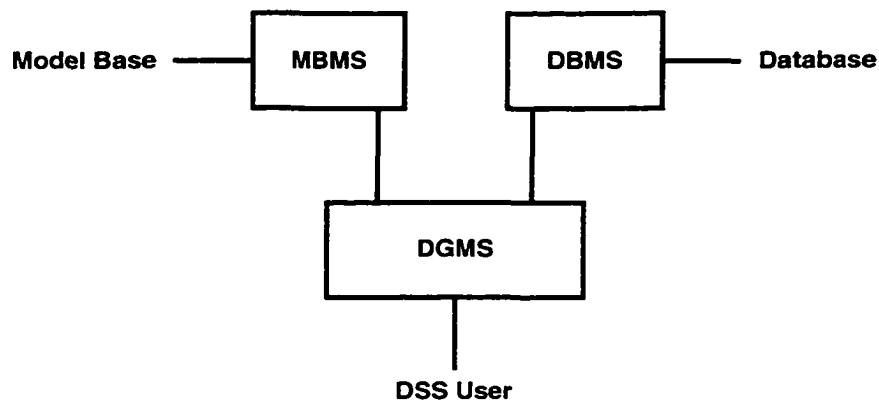


Figure 4.2: DSS components after [Sage 91]

Therefore, I decided to build up a simpler, but more useful model that allows for designing the relationships between different design parameters and the collection of data that can be used later in a more complete DSS. This decision set the focus of my research more on building an MBMS rather than the DBMS part of a DSS. As for the DGMS, I intended to set up a prototype that could be used for evaluating the *Interaction Constraint Model* on its applicability.

S-04: Deciding about the usage of speech (or any other interaction mode) to support a specific activity is a binary decision and thus depends more on the relationship of the influencing parameters than on their priorities or values.

→ The interaction model can be based on binary decision models that focus more on the relationship of constraining factors.

This model will serve more as an “interactive guideline” than a DSS. Guidelines are rules and instructions on how to do certain tasks. Some guideline on the usage of speech recognition are available [Rudnicky 1996][Sun 1998, p.19-33], but are not yet very complete and especially lack the problems that occur while using speech on mobile and wearable computers in changing environments.

One more advanced tool that helps system designers in decisions about using speech for input and output is described in [Bernsen 1999]. This research group collected and formalized a large amount of claims about the usability of speech technology in a multimodal environment from the literature [Disc 1999, p.13-74]. SMALTO is the “Speech Modality Auxiliary Tool” [SMALTO 1999]. It provides a collection of the “right” claims for a problem that can be approached from different directions. The system designer can choose to start with retrieving claims about the “application”, “speech input”, or “speech output”. The designer has to read these claims that the tool provides and has to decide whether the claims apply to the problem at hand and which conclusions to draw from them. The tool is based on a “Modality Theory” that defines 25 “Modality Properties” which are single or combined input and output modalities, such as “Linguistic Input / Output”, “Acoustic Input / Output”, or “Speech Input / Output”. This research group is convinced of the applicability of decision support tools for choosing the right interface modality: “that the applied theoretical approach adopted, which is based on Modality Theory, is sufficiently successful to warrant tools development” [Disc 1999, p.2]. However, this approach is not yet sufficient for the interaction design with mobile and wearable computers, since it does not reflect the different conditions in which the system might be used. Also, this research was targeted towards natural language recognition, which might be very much anticipated but not applicable in the foreseeable future, especially with the limited resources of mobile and wearable computers.

Keller and Hurri mention “state-based” requirements [Keller 1990] [Hurri 2000, p.31] that are focusing on the “states of nature” and the possible requirements options. If we apply this approach to “state-based” user interaction, we can set up the interaction model based on patterns, which could be identified by analyzing the “states” of the specific interaction activities. Or more specifically, assume the actual tasks that should be supported as the “states” and collect patterns of their properties. Design patterns offer a good level of “examples / guidelines” [Gamma 1995, p.26-28], which leads to the idea of collecting (speech) interaction design patterns that can be used as “examples / guidelines” for designing speech-enabled software applications.

S-05: Providing an interactive guideline based on patterns and illustrative examples, the latter in an implementation-independent form, will give a substantial guidance for interaction design.

→ If the interaction model can give templates for interaction patterns and the evaluation and test implementation of the system can show some real-world examples, it will be usable solution for interaction design.

4.4. Constraints Definition

Leffingwell and Widrig define constraints as “a restriction on the degree of freedom we have in providing a solution” [Leffingwell 2000, p.44]. They mostly describe constraints that are given at the design time, but not those that only occur at the time of the actual use of the software. The latter are particular interesting for interaction modeling since interaction is highly constraint by the interacting parties and the environment in which the interaction occurs. An INCOSE working group goes a bit in that direction in stating that “constraints [describe] on how and where the other requirements apply, or how the work is to be performed” [Caple 2001, p.4-5]. They proceed with arguing that “most process requirements tend to be constraints” and that some service requirements “may emerge during project as a result of constraints”. Herein, process requirements are defined as “requirements for how the work is performed” and service

requirements as “requirements for the services to be provided”. This leads me to the following assumptions:

- Requirements Engineering focuses on design time
Example: The device shall store 20 inspection reports!
- “Constraints” on the (speech) interaction model are requirements at operation time
Example: climbing bridge requires both hands;
Example: heavy ambient noise disallows use of speech recognition.

It is important to care about the constraints because requirements act differently in different situations and in different combinations. Thus, we have to get a clear vision of what happens at operation time. To get this vision, we first have to decide on the underlying model of the *constraints* and their interactions. Thus, the first step is to define the different components or *categories of constraints*. These categories have to map the needs of describing the *constraints* at a specific situation at a specific time like a snapshot of every system component that is active - or “influencing” even through being not active - and thus have to represent the “stakeholders” of the interaction situations. I give a definition of *constraint categories* in Chapter 5.1.

S-06: Regarding the constraints at operation time and especially the relationship between these constraints is essential for providing the right interface at the right time.

➔ This will give an additional perspective to requirements analysis of the traditional (non-mobile) computing systems.

5. The *Interaction Constraints Model*: Description

In the following chapter, I will describe the composition of the *Interaction Constraints Model*, and the research path that I took. First, I will describe the components that compare the interaction model and the intensions for each of the different components, and then I will explain the modeling techniques that I chose. Finally, I will give an overview of how the model might be used. This chapter concludes with an evaluation of the model.

5.1. A Constraints Model

Before the introduction of mobile computing, the requirements of IT systems could be associated with mainly three categories, namely “User”, “Device”, and “Application” Now there are five, since mobile and wearable computers implicate changing tasks in changing environments. These “*constraint categories*” contain *constraints* that interact during operation time.

5.1.1. Task

Tasks are considered to be “states in the working process” as a part of the workflow, i.e. the temporal relationship between single *Tasks* plays a secondary role. This implies that e.g. each “rounded rectangle” in a UML Activity Diagram will represent a unit for *constraint* evaluation.

5.1.2. Environment

The considered *Environments* are those, in which different (non-traditional) input modalities are applicable and different demands on the user are present (e.g. office environments are covered by existing HCI research).

The *Environment* is furthermore defined as the working / usage environment of the device, composed of such influences as ambient noise level, lighting, potential hazards (need for gloves, masks, etc.). However, properties of the IT infrastructure are covered by the *Device* description.

5.1.3. Application

The *Application* influences the user interaction by demanding different navigation / operation *Tasks* of the software, e.g. a CAD application deals with 2D or even 3D drawing navigation whereas an inspection application deals more with check lists. There are domain-specific applications, such as construction or manufacturing applications and general applications that for example support the “back office” processes. Furthermore, different application structures or software architectures cause different behaviors of the software. Finally, the *Application constraint category* holds the actual interface / interaction layer, i.e. the interface to the user of the device.

5.1.4. User

Users have different cognitive, logical, and physical abilities, as well as different expertise and experience. Users and their capabilities are also constrained by the working environment, such as situations that demand special attention or occupy the *User* in some way. Since the focus of this research lies on the investigation of possible interaction modalities in a given *Environment* for a specific *Task*, the actual *User* preferences are covered by providing all possible interaction modalities to the *User*, who then can decide at operation time whether to use one of the possible modalities.

Working habits of users should not primarily go into the *constraints* design, since these habits might change completely with the use of the mobile IT support. However, these habits have to be investigated thoroughly to fully understand the *Tasks* that have to be support by IT.

5.1.5. Device

The *Device* is considered to support speech technologies, i.e., to run a speech recognition engine. This is a not too much anticipated assumption, since even PocketPCs that are amongst the smallest hand-held devices at the time of this research are now capable of

running speech engines. However, different self-contained or client/server architectures will be considered as *Device constraints* (see *Application constraint*).

The *Device* (which also includes other devices in reach by the *Device*) can offer different input / output modalities that may or may not support speech or interfaces that are more appropriate for a given *Task*.

S-07: Classifying the constraints into a 5-component constraints model provides a clear basis for setting up an Interaction Constraints Model.

→ These 5 categories sufficiently cover the *constraints* that are necessary to describe interaction situations and the relationships that exist between these *constraints*.

5.2. Task definition

The activities that have to be observed and analyzed to appropriately design user interaction are the actual work tasks that the worker has to “get done”. Thus, I focus primarily on *Tasks* that involve both activities for the “actual job” and for interacting with the device.

The following sections describe how I subdivide *Processes* into *Tasks* and *Steps* and how I categorize *Tasks* in respect to their level of interaction between human and machine.

5.2.1. Processes, Tasks, and Steps

From workflow management, I use some of the analysis methods and parts of the terminology. I describe the work that should be supported by mobile IT support in a 4-level terminology: “Process Chain” → “Process” → “Task” → “Step”. A *Process* is a portion of work that workers perform with a certain high-level goal, such as changing the spark plugs of a vehicle engine. A *Task* for me is the portion of a *Process* that an experienced worker would not usually interrupt unless an unexpected event occurs. For example loosening a single spark plug would be such a *Task*. Unless the spark plug is too tight such that the technician has to get a special tool,

the technician would loosen it in one continuous flow of activity (or *Task*). The single *Steps* that, in this example, would combine to form the *Task* “loosen spark plug” would be grasping of the wrench, placing the wrench on the spark plug, turning the wrench, etc. A set of *Processes* that a worker would perform during a specific time is called a *Process Chain*. An example for the *Process Chain* is the complete diagnosis of an engine, where the change of the spark plugs is a single *Process*, resulting from the *Process* “check spark plugs”.

The UML only defines the term “Activity”, which I see as an abstraction of *Process*, *Task*, and *Step* and thus use during the modeling of use cases and scenarios. Figure 5.1 shows a UML activity diagram with the described decomposition into *Processes*, *Tasks*, and *Steps*.

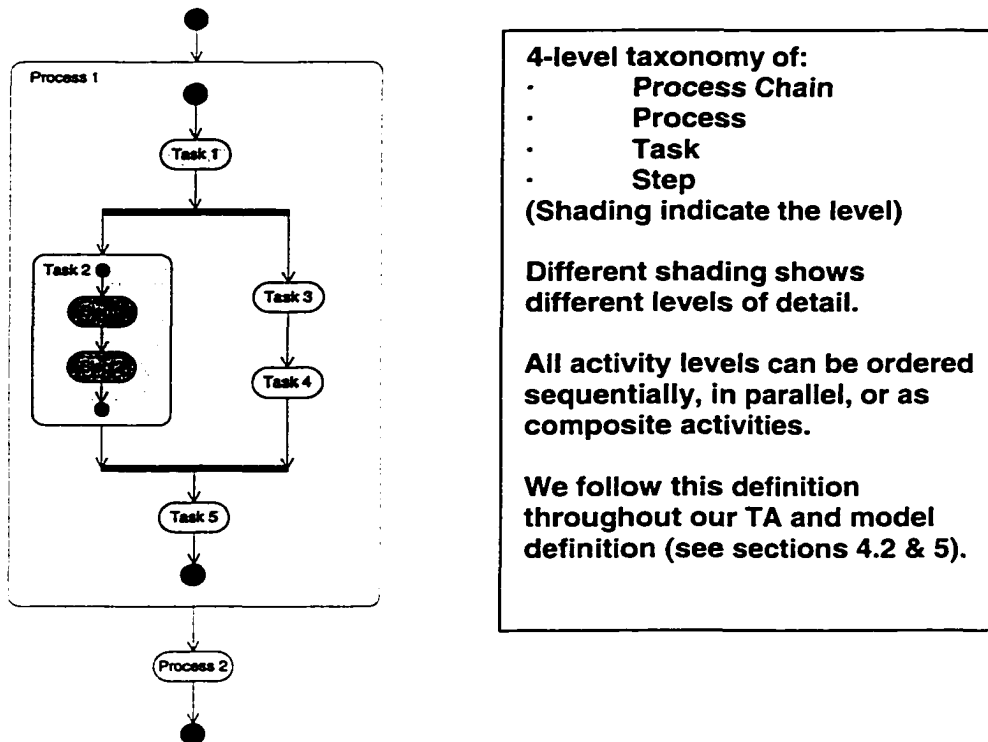


Figure 5.1: 4-level terminology shown in a Composite Activity Diagram

I am aware that these are different terminologies from those used in HCI and SE, but since I am targeting industrial applications, I have chosen to use these workflow terms to better represent the primary activities, the workers actual job, so that I can better map a generic speech interaction model to this workflow. Also, the goal of this research is not to provide detailed

implementation guidelines, but rather to help in the decision about the appropriate and effective interaction modes for a specific application. The detailed implementation of the speech interaction can then be described in the terminology preferred by the software development team.

5.2.2. Interaction levels of Tasks

The first step to design an IT system that supports workers at their workplace is to look at their actual work and work environment and then find ways to incorporate the IT support into this workflow. The following is a description of seven different *Task* categories that are defined by their level of interaction between the *User* and the *Device*. There are three basic categories of “**Independent Tasks**” that can be identified. Additionally, there are four “**Composite Tasks**” that are built by combining tasks of the three basic categories. According to Table 5.1, the *Tasks* are numbered relative to their position in the table:

- **Primary Task (PT):** No interaction with the device; i.e. no IT support is needed and applied.
- **Support Task (ST):** Sole interaction with the device and the device supports the user in providing information or accepting input; i.e. “productive” steps are done; e.g. reading a manual, or entering inspection results.
- **Control Task (CT):** Sole interaction with the device but task only involves navigating the software; i.e. no “productive” steps are done; e.g. scrolling a page or opening a file.

Task Category	PT	ST	CT	relevant
1				No
2				No
3				No
4				No
5				Yes
6				Yes
7				Yes

Table 5.1: Task Categories in terms of relevance for interaction with mw-CAE Systems

Only the *Composite Tasks* No. 5-7 (see Table 5.1) are considered important for this research, since the other categories (*Independent Tasks* and *Composite Task* No. 4) are irrelevant or covered by existing HCI research [Kieras 1997]. This is caused by the non-existence of computer interaction (No. 1), or by the independence from the *Primary Task*, which allows for focusing on the interaction with the device, although it might not be possible to change the environment completely, i.e. some of the *constraints* of the preceding *Task* will remain. In the task analysis, I would thus consider tasks, initially classified as No. 2-4 *Tasks* that are constrained by succeeding or preceding *Tasks*, as the corresponding *Task* with PT involvement. For example, if a *Task* only involves a *Control Task* (No. 3), but occurs on scaffolding in high ambient noise caused by a preceding inspection task at a bridge, this *Task* would become a No. 6 *Task* with the *constraint* imposed by the preceding *Primary Task* (inspection on scaffolding).

S-08: Orienting the task analysis at the actual work tasks reflects the actual constraints implied by the task and the environment at operation time.

→ To get the true *constraints*, we have to look into the actual work activities of the User.

5.3. Modeling Techniques

A good description of a system can only be as good as the techniques used in this description. I chose the Unified Modeling Language (UML) for the following reasons:

- The UML has become a quasi-standard in object-oriented modeling [Rumbaugh 1991] and thus is understood by a broad audience. Furthermore, it has an easily understandable notation, which allows for a quick reception of the described concepts and models.
- The UML is capable of describing all the involved parts, like task analysis (Activity Diagram), user interaction modeling (Use Cases and swimlanes in Activity Diagrams as means of assigning roles), interaction between system components (Packages), and pattern design (e.g. Class Diagrams).

- The UML allows for automated generation of source code (e.g. Java, C++, etc.) or the translation into other notations (such as XML or entity relationship models) with the help of CASE tools (Computer-Aided Software Engineering tools). This enables an iterative design process. In my research, this will be useful to fine-tune the developed model when I validate it with data from real-world examples.
- We can find a lot of work that is also described using the UML, which facilitates creating the models in my research on this existing research rather than modeling from scratch. Thus, I was able to derive some ideas or concepts more easily from sources such as Gudgeirsson [Gudgeirsson 2000] and Lif [Lif 1998], who designed their research and documentation using the UML.
- Finally, the UML also helps in the very much appreciated use case-based design process. Ivar Jacobson introduced use cases in 1992. Jacobson is one of the key people in object-oriented software design [Kruchten 2000, p. 98] [Jacobson 1992]. Use cases allow for a graphical representation of system functionality and the participating “actors”, such as the users of the system or other systems. In [Van Harmelen 2001, p. 161-243] Kruchten explains how the user interface design is handled in the Rational Unified Process (RUP). The RUP is a concept developed by Rational Software [Rational 2001] [Kruchten 2000] to ensure a holistic design and engineering process for (software) products.

The outcome of this research will be an *Interaction Constraints Model* that describes a) the means of interaction between a user and a mobile or wearable IT device, and b) the *constraints* on this interaction. Thus, I will describe this model in two separate ways: the interaction model and the *constraints* as attributes of the different model components. However, identifying these attributes and describing these attributes in an unambiguous way are two very different tasks. Thus, Sections 5.4 and 5.5 will describe the model itself and Section 5.6 will discuss the attributes of the model components.

5.4. User Interface Modeling

In [Lif 1998, p. 89-99] the author describes User Interface Modeling (UIM), a technique to map requirements that occur in different “work situations” to user interface elements and their components with the use of “workspaces”. In an earlier publication, he and the coauthors define a work situation as “a set of related work tasks without sequential restriction, but with a natural belonging performed in total by one person. It corresponds to a workspace in the user interface. One work situation may include one or several work tasks. One actor can handle one or several work situations.” [Gulliksen 1997, p. 282]. The concept of workspaces is to offer all the needed information of a specific work situation in one place to allow users to directly access this information. A work situation is hereby mapped to a specific workspace, which covers a set of “use cases”. These use cases represent the single tasks that a user has to perform. Thus, use cases illustrate the functionality a software application should offer to support the user in performing these tasks. Each use case, or each piece of software functionality is realized by (user) interface elements, such as tables, charts or buttons. The concept of UIM is intended to facilitate the design of user interfaces, based on requirements that are identified during task analysis and requirements elicitation.

In this research, I took a similar approach but instead of defining workspaces as mainly GUI elements, I defined them as a set of possible user interface components, such as graphical input and output, speech recognition and synthesis, or gesture recognition. Lif identifies work situations based on the actor within this work situation. This means that the actor and his goals are sufficient to identify the requirements for a specific workspace. This is true for systems that are only used in one specific environment and that support an activity that is mainly performed using a computer (supporting tasks of categories 2-4 defined in Table 5.1). However, for the design of interaction with mobile and wireless computing devices the work environment and the Primary Task, the activity that the device is intended to support, not primarily the actor, influence decisions about applicable interaction means. Therefore, I added these two factors to my definition of the work situation.

Figure 5.2 shows this derived model. In the first level, “actors”, “work locations” and “work activities” define certain “work situations”.

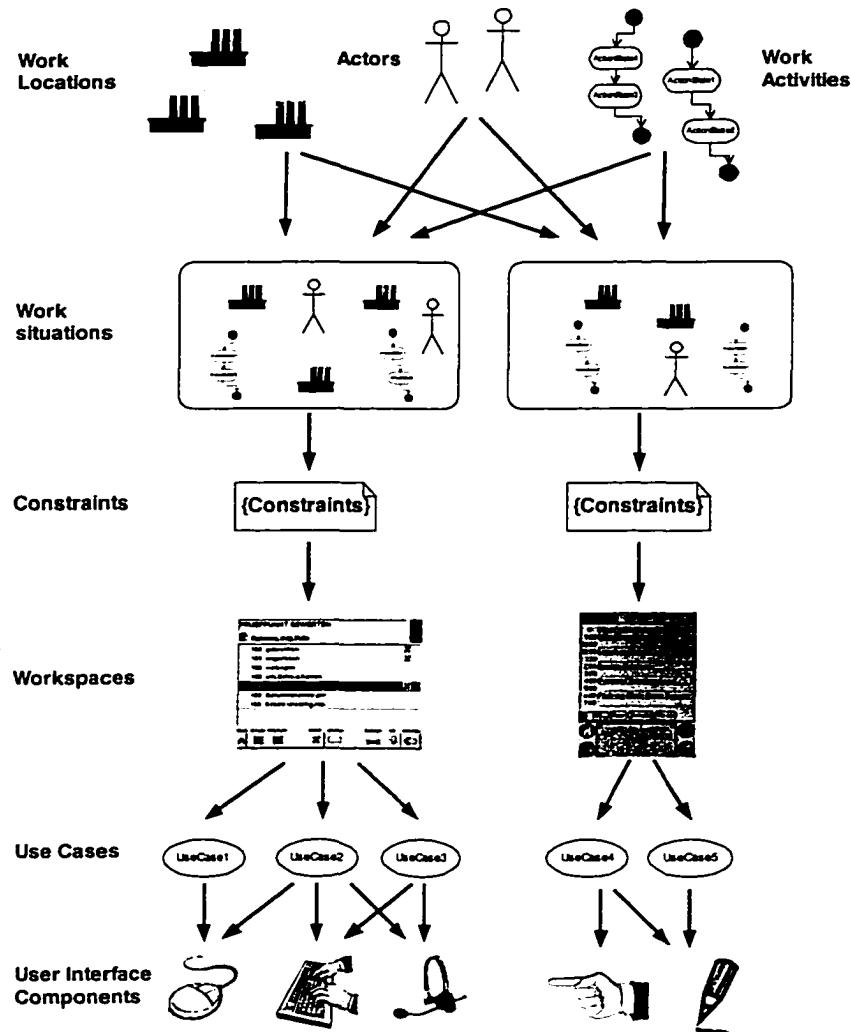


Figure 5.2: User Interaction Modeling derived from [Van Harmelen 2001] and changed to meet the needs for the *Interaction Constraints Model*.

Each **work situation** should be supported by appropriate means of interaction. Herein, **work locations** describe the conditions that surround the actor in this **work situation**. These are for example physical conditions, such as lighting or ambient noise, as well as the presence of additional IT infrastructure or different support systems. **Work activities** are *Primary Tasks*,

according to Section 5.2. that have to be supported by the computer system. These activities set the required functionality of the system but also the limitations caused by the workflow or demands of these activities. **Actors** represent the link between *work activities* and *work locations*. The *actor* has no particular influence at this level of the model, but instead plays a role in defining the “constraints” that are caused within the work situations.

The **constraints** from these situations that influence the interaction between the user (the *actor*) and the mobile or wearable computer system are captured and define the properties and functionality of usable “work spaces”. **Workspaces** in my model define a set of possible user interaction means. This means that generally, all interaction means are possible as long as they are not ruled out by *constraints* from the *work situations*. In other words, in an unrestricted *work situation*, the user can choose between all available interaction means, i.e. from any implemented interaction modality. Should there be any restrictions caused by some conditions or *constraints* of the *work situation*, some or even all of the interaction means are ruled out, i.e. should not be enabled by the system.

Different **use cases** describe the interaction functionality that is offered by a single *workspace*. Each use case describes a generic action of the user, such as entering text or filling in forms, that can be performed using different “user interface components”. The **user interface components** define the way a user interacts with the computer system. The usability of a particular user interface component is dependent on the *constraints* of the *work situations*.

Based on this model, it is my hypothesis that by analyzing previous projects, we can identify successfully implemented and applied *user interface components*. Furthermore, we can identify the software requirements that led to these successful interfaces. Simultaneously, the *constraints* that make certain interaction means a bad choice become clear. Another hypothesis is that we can transfer these *constraints* to similar *work situations*, and facilitate and speed up the decisions about which *user interface components* to use for specific *work situations*.

5.5. Model Composition

The intention of the *Interaction Constraint Model* is to have a taxonomy or a high-level description methodology for documenting the *constraints* for the use of (speech) interaction to support a specific activity, at a specific location under specific conditions / *constraints*. In reusing this documentation and reviewing the design and implementation decisions of other applications, we can base our design decisions on previously made experience. The concept is to capture “snapshots” of work situations – as discussed above - which occur at operation time of the mobile support systems and then consider and evaluate possible user interfaces for the specific interaction. A work situation contains three main parts (see Figure 5.3):

- 1) The **Constraint** object captures the actual *constraints* caused by different influences of the *work location* and the *work activity*;
- 2) The **Work Situation** is composed by the *work location* and the *work activity*. Together, the latter two define a unique *work situation*;
 - 2a) The **Work Location** object contains information about the conditions in which a *work activity* is conducted;
 - 2b) The **Work Activity** object, which represents the Primary Task (see Section 5.2), is supported by the mobile device; and
- 3) The **User Interface** object, which will be used to document information about the actual implementation of specific interaction modalities.

Figure 5.3 shows that *constraints* are linked to the *work location* and the *work activity* and thus define a unique *work situation*. It also shows that *constraints* are connected to user interfaces. Together, this presents how the *constraints* of specific *work situations* influence the design of the user interface.

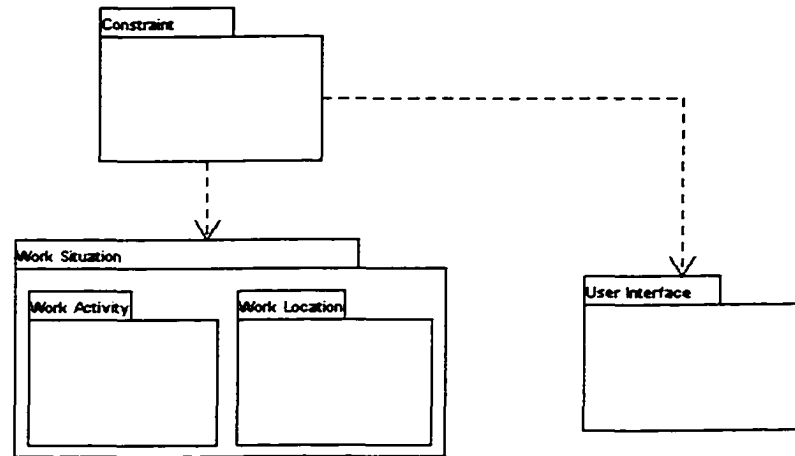


Figure 5.3: High-level composition of a *constraint / work situation* model

In the following sections, I will describe in detail how these components fit together and build the model that allows the collection of *constraints* and their influence on the user interface design.

5.6. Constraints

The *constraint* object is the essential component of the model. It contains the information about what the nature of the *constraints* is, such as a not readable display, and the *influence* from a specific *work situation* and a specific *work activity*, which describe the actual cause of the *constraint* (see Figure 5.4). A *constraint* can be defined as “a restriction on the degree of freedom we have in providing a solution” [Leffingwell 2000, p.44]. This “solution”, in the case of the *Interaction Constraint Model*, represents the applicability of certain user interaction means. I categorize *constraints* in five groups (see Section 5.1): *User*, *Environment*, *Task*, *Application*, and *Device*. In mapping the *constraints* to these categories, the resulting restrictions on the user interface design become more obvious and reproducible. This mapping allows for example to map the *influence* of “user shall wear protective gloves” to a *constraint* of “user’s sense of touch is restricted”.

Constraints also map to *requirements*. Leffingwell defines (software) requirements as follows: “Software requirements are those things that the software does on the behalf of the user or device or another system” [Leffingwell 2000, p. 229]. The author proceeds with “The first place to look for software requirements is around the boundary of the system for things that go ‘into’ and ‘out of’ the system: the system interactions with the user.” Thus, in setting up *constraints* for user interaction, we can derive the requirements for this user interaction. This may result in some restrictions or variations of the implementation of the interaction mode, or even in the interaction mode being not applicable.

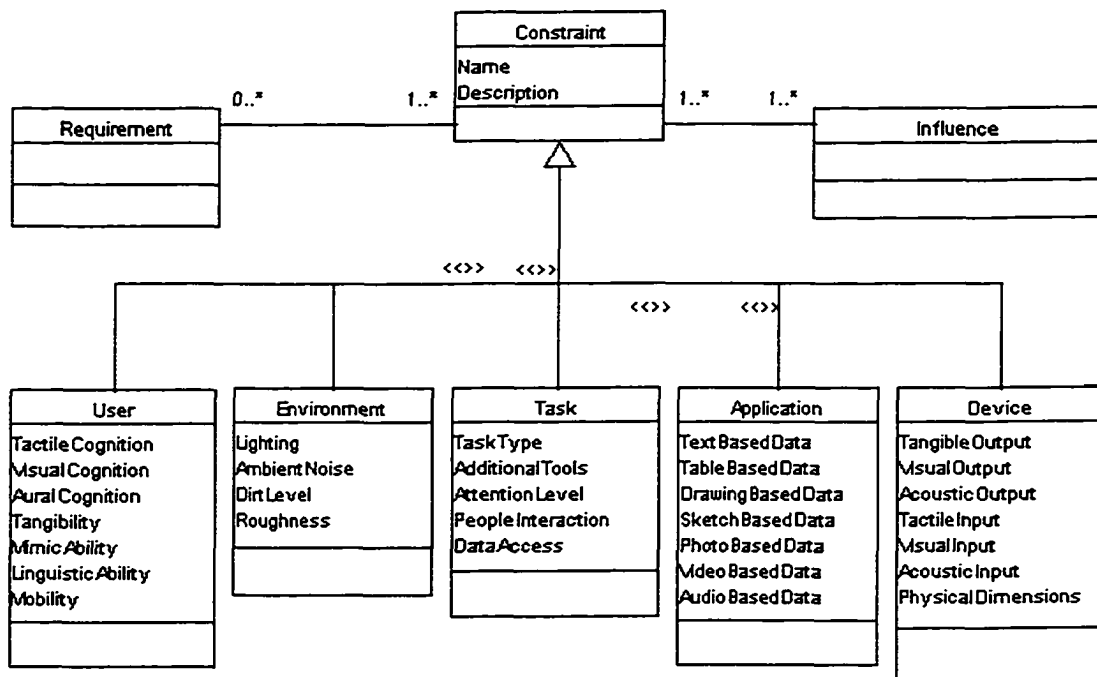


Figure 5.4: Composition of the *constraint* object.

How to define and manage requirements is part of the particular design team's design concept or design process. Thus, I keep the requirement object as a placeholder for future integration with requirement management tools. Two examples of managing requirements are the Rational Unified Process, which gives a general approach to requirements management

[Kruchten 2000], and that by Gudgeirsson, which uses an XML-based approach to represent and manage requirements [Gudgeirsson 2000].

The level of detail of the *constraints* collection has a great impact on the resulting data quality. If the level of detail is too low, i.e. there are too few attributes for each *constraint*, the data might not be meaningful enough. However, if there are too many details, i.e. too many attributes for each *constraint*, the collected data might not allow designers to find any congruent *constraint* patterns. For the same reason, I chose limited value ranges for each *constraint attribute*, i.e., mostly two or three options per value, instead of a broad range of values. Thus, I defined the attributes for the five *constraint categories* as follows. An example at the end of each description will illustrate how these attributes will be assigned.

The following descriptions show that some attributes of different *constraint categories* are similar, which results from the fact that the single categories are not directly related to each other. However, the *constraints categories* can show contradicting entries that have to be resolved during the design process. For example the linguistic cognition of a user, which represents the ability of a use to enter speech, can be restricted by too much ambient noise, e.g. on a construction site, or by the fact that the user walks through a museum, where silence is expected. Thus, we cannot assume that the value of an attribute of one *constraint category* implies a certain value of an attribute of another *constraint category*. This issue is also reflected in the *device* and *user constraints* that have corresponding but independent attributes. Although these attributes can have different values, the combination of them decides on the actual design *constraint*. For example, the device might not offer speech input, the user might be hindered in speaking, or the ambient noise might be too loud. Although these attributes are independent, any of these can disallow the use of speech input as an interaction mode.

Finally, the direction of the impact of the different *constraints* also varies: whereas *environment, task, and application* generate *constraints*, the user and the device are impacted by the *constraints* of their category. This reflects the fact that *interaction constraints* are oriented towards the user and the device, which are the peers of the actual interaction.

5.6.1. User

User constraints for the interaction with a device are typically input or output *constraints* that do not allow or at least restrict interaction through specific information channels. Information channels are in this case the human senses. Thus, we can identify the attributes that describe *constraints* on a *user's* information channels or human senses in a specific *work situation* in respect to the *user's* visual, aural and tactile input and output abilities. To not confuse these terms for input and output with the *device's* attributes, I define these as shown in Table 5.2. The three-level categorization of the possible values seemed to offer the best level of detail for the purpose.

Attribute	Description	Values
Visual Cognition	The user sees information.	"free visual cognition"; "visual cognition restricted"; "visual cognition blocked".
Aural Cognition	The user hears information.	"free aural cognition"; "aural cognition restricted"; "aural cognition blocked".
Tactile Cognition	The user receives information with sense of touch.	"no hands blocked"; "1 hand blocked"; "2 hands blocked".
Mimic Ability	The user transmits visual information, such as gestures.	"free mimical ability"; "mimical ability restricted "; "mimical ability blocked".
Linguistic Ability	The user transmits information, using speech or sounds.	"free linguistic ability"; "linguistic ability restricted "; "linguistic ability blocked".
Tangibility	The user transmits information with the sense of touch.	"free tangibility"; "tangibility restricted "; "tangibility blocked".
Mobility	The user's ability to freely move about the work location.	"complete freedom of movement"; "freedom of movement restricted"; "no freedom of movement"

Table 5.2: Attributes of User Constraints

Example: *High ambient noise can restrict or block the user's Aural Cognition and Linguistic Ability; carrying a flashlight blocks one of the user's hands; and a huge, heavy device restricts the user's Mobility.*

5.6.2. Environment

Environment constraints originate mostly from the fact that the mobile or wearable device is used under non-optimal conditions. This means, that for example the lighting on a bridge might be too bright, caused by an influence “sunlight”, and in a tunnel too dark, caused by the nature of a tunnel. Note that the sunlight is not the *constraint*, but the resulting bright light in the *environment*. The sunlight however, would be considered an *influence* in the *Interaction Constraints Model*. Table 5.3 shows a list of the most common attributes of *environment constraints*. Here, too, a three-level categorization is sufficient for proving the model. Future extensions will perhaps assign figures in Lux and decibel values for lighting and sound, or the IP rating for water and dust resistance. However, to collect these more accurate values, we would then have to use automatic light or acoustic sensors.

Attribute	Description	Values
Lighting	The level of lighting that occurs at the work location.	“Low”; “Normal”; “High”.
Ambient Noise	The level of ambient noise that occurs at the work location.	“Low”; “Normal”; “High”.
Cleanliness	The level of cleanliness, especially around the device and the user.	“Low”; “Normal”; “High”.
Harshness	The level of harshness, such as sudden drops that affect the device / the user.	“Low”; “Normal”; “High”.

Table 5.3: Attributes of *Environment Constraints*

Example: *Running engines in an automotive workshop can cause high-pitched noise and thus set the Ambient Noise to “high”; while pouring concrete, the cleanliness level of the construction site can be described as “low”.*

5.6.3. Task

Task constraints depend mainly on the type of the *task*, which I defined in Section 5.2. Other *task constraints* originate from the *task's* nature and the way it occupies the *user* or the *device*. For these decisions and due to the broad variety of additional tools, distractions or data access possibilities, I chose a Boolean (yes/no) decision for the evaluation for these attributes, which reflects that there are no possible intermediate decisions unless there would be an exact description of the needed tools, the attention level or the amount of data to be accessed. Future extensions of this model may define one or more of these values to address needs of detail that I could not foresee at this point. For example, the way people communicate with each other might change with the introduction of more and more powerful mobile and wearable devices. Table 5.4 shows the attributes, I defined for *task constraints*.

Attribute	Description	Values
Task Type	The combination of Primary, Secondary and Control Task.	"Type 1" – "Type 7", (according to Section 5.2)
Additional Tools	The tools the user has to carry, additional to the IT device.	"Yes"; "No".
Full Attention	The task might bind the users full attention and thus allows no interaction.	"Yes"; "No".
People Interaction	The task might demand interaction with other people.	"Yes"; "No".
Data Access	The task might need data that are accessible within the IT infrastructure.	"Yes"; "No".

Table 5.4: Attributes of Task Constraints

Example: *During a surveying job the surveyor has to carry a leveling board and other instruments, which would assign the Additional Tools to "yes"; if a task is difficult and does not allow the user to be distracted, the task needs the user's Full Attention.*

5.6.4. Application

Application constraints are constraints that are mainly based on the kind of data (representation) that has to be entered, accessed, or managed. The collection of these kinds of data will take place in observations of the user's actions in the task analysis, or during the analysis of existing software applications and legacy software. The attributes of the *application constraints*, shown in Table 5.5, are Boolean values to indicate the existence or absence of a specific kind of data (representation) in a *work situation*. During the assignment of specific values, we have to recognize new technologies and even new processes, enabled by these new technologies. Sketches, for example, become easier to start with sketch templates that are offered by the system. On the other hand, the handling of huge drawings is limited if small displays restrict the viewing area. Thus, some activities might make use of different kinds of data representation in the process without IT support or with older variations of IT support in contrast to the envisioned system.

Attribute	Description	Values
Text-Based Data	Text has to be entered or displayed (Word processing)	"Yes"; "No".
Table-Based Data	Data is stored or represented in table (Spreadsheets)	"Yes"; "No".
Drawing-Based Data	Data is captured or provided in graphical form (CAD)	"Yes"; "No".
Sketch-Based Data	Data is in captured or provided in simple sketches (Sketch design tool)	"Yes"; "No".
Photo-Based Data	Data is captured or provided as realistic views (Picture / image viewers)	"Yes"; "No".

Table continues on next page.

Table continued from previous page.

Attribute	Description	Values
Video-Based Data	Data is captured or provided as moving pictures (Video capture / player)	"Yes"; "No".
Audio-Based Data	Data is captured or provided acoustically (Sound recorder / player)	"Yes"; "No".

Table 5.5: Attributes of Application Constraints

Example: *If an application needs the functionality of sketch-based data, such as to point out the location of a crack in a bridge component the Sketch-Based Data attributes is set to "yes"; similarly, the Audio-Based Data attribute is set to "yes", if short text messages from an inspector should be captured as audio files.*

5.6.5. Device

Device constraints are similar to the *constraints* that affect the *user*, just at the other end of the information chain. Similar to the *user* side, we can define visual, aural and tactile input and output channels. In the following list, the terms in parentheses are the counterparts in the *user constraints*. The interaction capabilities of a *device* are more error-prone than the capabilities of the *user*. This is based on the lack of adaptivity and anticipation the device can offer. A *user* can more easily compensate a lack of a piece of information. Thus, I took Boolean values rather than the three-level categorizations for the input and output attributes at the device side. In that way, the information channel of the device is either enabled or disabled. When devices become more capable and more intelligent, these attributes can be adapted. For the last attribute, the physical dimensions, I decided to take analogies that are common for designers of mobile and wearable computers, since I am not convinced that actual numbers would help too much in this case. I

believe that mapping *devices* to classes of *devices* with similar dimensions is more appropriate here. Furthermore, these analogies can be easily translated into the specific values for size, weight, and volume of the device. Table 5.6 shows the attributes of *device constraints*.

Attribute	Description	Values
Visual Output	E.g. Graphical User Interface, LEDs (Visual Cognition)	"Yes"; "No".
Acoustic Output	E.g. beeps, text-to-speech (Aural Cognition)	"Yes"; "No".
Tangible Output	E.g. vibration alarm, tactile display (Tactile Cognition)	"Yes"; "No".
Visual Input	E.g. gesture recognition (Mimic Ability)	"Yes"; "No".
Tactile Input	E.g. buttons, pointing device, touchscreen (Linguistic Ability)	"Yes"; "No".
Acoustic Input	E.g. speech recognition (Tangibility)	"Yes"; "No".
Physical Dimensions	The size, weight and volume of the system in respect to its usability.	"N/A"; "Phone size"; "PDA size"; "MA IV size"; "Clipboard size"; "Laptop size".

Table 5.6: Attributes of Device Constraints

Example: *If the device that is intended to use for a specific application does not offer any sound output, the Acoustic Output attribute is set to "No"; if the considered device is the size of a PDA, the Physical Dimensions attribute is set to this value.*

Since all 30 attributes of the five *constraint categories* and their possible values are technically independent, the *constraint space* that is covered, or the different possible combinations of attributes that a *constraint pattern* can describe, is very large, i.e., about 975 billion possible different *constraint patterns*. However, if when matching *constraint patterns*, minor

variations in some of the *constraint attribute values* are tolerated or if we focus the search only on specific parts of the *constraint pattern*, we can find previously addressed *work situations* even from this large *pattern space*.

As the results in Chapters 8 and 9 demonstrate, I could indeed find matching *work situations*, mainly caused by two reasons: similar patterns occurred within the investigated projects and a some repeating patterns occurred within the attributes *constraint categories*. The first reason, the similar patterns occurring within the projects, reflects the fact that although *work situations* vary within a project, they still contain similarities in their *constraint patterns*. On a construction site, for example, one will always find *work situations* with low cleanliness and some construction machinery running. The second reason, the repeating patterns occurring within *constraint categories*, results from the fact that although the attributes are independent, some combinations of attribute values will be more likely than others. For example, high ambient noise will most likely influence the linguistic ability of the user. However, since the attributes are independent, the model also covers the surprising results that are not easy to imagine.

The large *constraint space* also requires this approach to have a significant number of projects in the database to truly achieve productive results for the design process. Also, the intention of this approach is not to cover all possible design problems, but rather to provide quick advice on more common questions that could be answered from the wealth of experience that is being collected as these systems are designed, implemented and tested. The model becomes more knowledgeable, and thus more useful, as more *work situations* from different projects are entered. If the system will be used only in a special context, e.g. for certain types of devices or application, the compilation of this knowledge will be faster.

5.7. Work Location

Work locations identify the location and thus the conditions, in which a *work activity* is performed. In this model, *work locations*, together with *work activities*, define a unique *work*

situation. The reason to have locations as an identifying factor in the model is to distinguish between different locations within one project and amongst different projects. Having said this, the actual goal after running several projects and collecting data from each project, is to find patterns that identify similar or even identical locations between projects, and thus to re-use the identified *constraints* on the user interfaces for each location (pattern). This will enable a learning process and a knowledge base for better-informed interface decisions. Since all the relevant information is related to *work situations* and stored as attributes of *constraints*, the *work location's* attributes sufficiently described with a name and a short description to capture the nature of the *work situation*. In fact, the concept of the *Interaction Constraints Model* is to analyze the data of the combinations of *work locations* and *work activities* and not the single entities.

Example: *“Inspecting a bridge structure” and “assembling tubular steel scaffolding,” have many conditions in common; e.g., the sunlight, the height of the workplace, safety concerns, etc. Working in a “tunnel construction” and in a “pit of an automotive workshop”, also has similarities: the artificial light (if any) and the dust / oil of the machines or vehicles.*

5.8. Work Activity

Work activities represent *Primary Tasks* (see Section 5.2). *Primary Tasks* are the tasks that the envisioned mobile or wearable device will finally support. As mentioned above, *work locations* and *work activities* define unique *work situations*. And the motivation for including the *work activity* as an identifier is similar as with the *work location*: Here, too, the goal is to find patterns of similar *constraints* that result from different activities and to re-use these patterns for design decisions of interaction in new *work situations*.

It may seem hard to compare activities from different domains and to find similar patterns amongst them. But the *work activities* themselves will not be compared, but rather the *constraints*

and the *constraints' influences* on the user interaction, which originate in these *work activities*. Thus, we loosen the *constraints* from being domain-specific and enable a domain-independent model.

Similar to the *work location's* attributes, also the information about activities is stored in the attributes of the *constraints*. Thus, the representation of the *work activity* is sufficiently covered with name and description attributes. The *work activity* should also not be seen as individual component of the model but as the second identifier of *work situations*.

Example: *It is obvious that some activities, such as "inspecting bridges" and "inspecting vehicles" have similarities, but even the "inventory of construction material" and "quality assurance at a manufacturing facility" can be mapped to a common pattern.*

5.9. Work Situation

Work locations and *work activities* define unique *work situations*. The link between the location and the activity is the user (or the actor, according to Figure 5.2) who literally brings the *work activity* to the *work location*. This is a new aspect that is caused by having mobile and wearable computers, which enable IT support away from the desktop or kiosk-like terminals. Thus, we have to identify varying sets of conditions or requirements, to which the design has to be adapted, and to which future adaptive devices will adapt automatically.

Each *work situation* is unique in a sense that exactly one *work activity* is performed at one *work location*. However, the conditions at different *work locations* and the demands of different *work activities* can have common patterns, and can thus lead to similar *constraints* on the user interaction with a mobile or wearable device.

In the model, *work situation* is an abstract object that is comprised of a *work location* and a *work activity* and has no actual representation. Although it does not have a representation, a *work situation* is described by the attributes of the *constraints* that are linked to it.

Example: We can use the two examples above to show the concept of a work situation: “bridge inspections” and “vehicle inspection” differ in their location; so do “assembling steel scaffolding” and “quality assurance” in respect to the activity. However, “inspecting a bridge’s interior structure” and “assuring the product quality in a badly lighted manufacturing plant” have similarities in both respects.

5.10. User Interface

The *User Interface* is intended to document information about the actual implementation of specific interaction modalities. Figure 5.5 shows the separation into different levels of information (the description of the interface itself, some implementation notes, and project information) that can be filled with data according to the status of implementation or the disclosure level of the project.

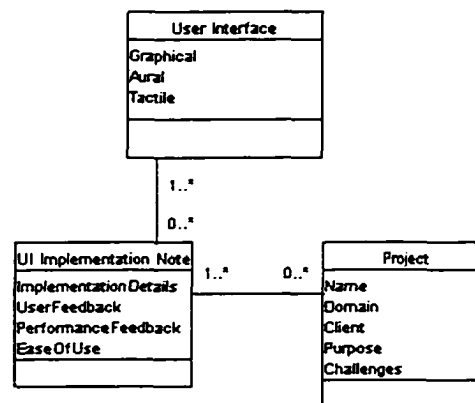


Figure 5.5: User Interface model

The *user interface* section is the place where the “lessons-learned” and the user feedback can be traced. To analyze, how users received specific interfaces components in a similar *work situation*, it is important not only to know which user interfaces have been chosen and implemented for specific *work situations*, but also to see and evaluate how these performed. However, as in all documentation processes, this is an additional workload, and only if we can

prove advantages from this additional work, either in terms of timesavings or quality improvements, we will end up with a usable *Interaction Constraints Model*.

5.11. Use Cases in the *Interaction Constraints Model*

Use cases help to define tasks or a set of tasks in an abstract way. They are a generalization of several scenarios that can be seen as instances of use cases. Another view would be to see each use case as a specific functionality of the system. As mentioned above, use case modeling found its way to the Rational Unified Process (RUP) and thus into the Unified Modeling Language (UML).

Choosing use cases as the representation method has two advantages resulting from the UML and RUP concepts. First, use case modeling supports the separation of the domain model and the interaction model [Kruchten 2000, p. 143]. Second, the possibility of representing relationships between use cases enables a kind of inheritance that can be used to build an API-like framework. In that way, we can build up the “interaction” or “system” use cases and integrate them into the “domain” or “business” use cases by defining UML “include” relationships. Fowler states that an “include relationship occurs when you have a chunk of behavior that is similar across more than one use case” [Fowler 2000, p. 44], which is exactly what I intend to do with the *Interaction Constraints Model*.

The concept I use is not to completely represent the *Interaction Constraints Model* as a use case-based hierarchy, but to apply the idea of identifying common attributes and separating these into different use cases or speaking with the *Interaction Constraints Model* separating and relating the common attributes to different *work situations*.

Through use cases, we can integrate the domain-independent *Interaction Constraints Model* into the domain or application model by including the interaction methods into the “View” layer of the “Model-View-Controller” software architecture. Figure 5.6 shows an example on how use cases can be shared between two projects from different domains, namely construction

management and automotive on-road service. The *work situations* of the automotive technician and the construction manager are similar, since both have to identify objects and both have to perform their jobs in an outside environment. However, in using the *Interaction Constraints Model*, we are not particularly interested in sharing the functionality of an object identification (which will be one of the future research topics), but the *constraints* that can be mapped to this use case (or *work situation*) and thus to the corresponding user interface.

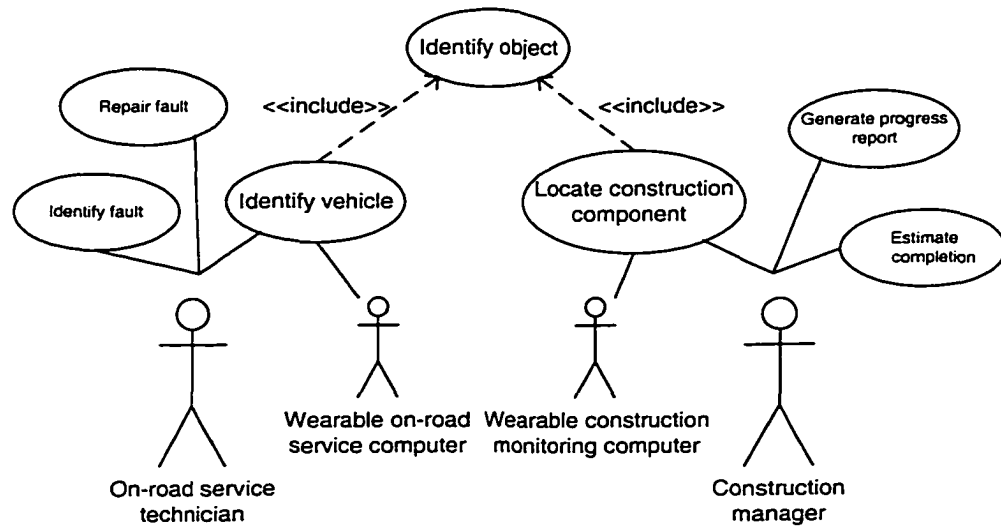


Figure 5.6: “Application” use cases that “include” the generic interaction use case “Identify object”

In the example above, the common *constraints* that are defined in the model, can be “sunlight” or “noise due to traffic / construction activities”. This set of *constraints* can thus be related to two *work situations* each defined by a *work location / work activity* combination: “identify vehicle” on “highway” and “locate construction component” at “bridge abutment”.

5.12. Evaluation of developed model

The relevant objectives to evaluate this model against are the “Secondary Objectives” identified in Chapter 3.1. In this section, I repeat these objectives and explain how I met these

objectives and which problems occurred during the modeling phase. Chapter 9 contains a more detailed evaluation on all objectives.

*O-04: The system has to be **domain-independent** (domain-neutral).*

As described in Section 5.9, the model was designed to meet this objective. In putting the constraints in the focus of analysis of work situations, not the data collection, but the data analysis can be domain-independent. Thus, we can still collect specific, detailed data, but then derive the set of data (the *constraints*) that is necessary to use the model.

*O-05: The system has to be **implementation-independent**.*

This model is not restricted to one specific implementation method. I designed it using the UML, and so far rather focused on the process description, rather than on implementing a proprietary system. This model, as will be described in the next chapter, can be incorporated in different kinds of decision support systems (DSS), and integrated with entity-relationship models, XML-based applications, or used in commercially available DSS.

*O-06: The system has to be **applicable to multi-modal interfaces**.*

I purposely kept the model open for different user interaction modalities. This enables a user of the model not only to answer my original question of “when to apply which speech interaction, and how”, but also to easily extend the model and use newly developed user interfaces that surely will be developed in the future.

*O-07: The underlying data model has to be **generic** enough to fit a significant set of collected data (requirements or constraints), but **narrow** enough so as not to be too fuzzy.*

To evaluate this objective is not completely possible at this point, because I did not yet put any data into the system. However, the developed model recognizes (as described in Section 5.4) the right level of detail in providing a manageable but meaningful set of attributes for each *constraint category*. The evaluation of this objective is part of the final evaluation (see Chapter 9).

*O-08: The underlying data model has to be **understandable by domain experts** but also **machine-readable** for future use in adaptive interfaces.*

This last objective cannot be evaluated completely at this point, but I would like to refer to the possibility of implementing this model as an XML-based data structure. This will enable – if implemented correctly – a data model that can be viewed and edited by domain experts, at least with some smart XML editing tools, and also be processed by XML parsers integrated in the application.

6. The *Interaction Constraints Model*: Implementation

To use and test the model that I described in the previous chapter, I implemented it in a system that allows for entering data and running queries to find the common *constraint patterns*. This implementation is not a complete decision support system, but is intended to prove the applicability of the *Interaction Constraints Model* for future decision support systems. Generally, there are several possibilities to approach an implementation. I will describe the most applicable implementation options and explain their advantages and disadvantages. Finally, I will state why I selected a relational database for the evaluation process. The evaluation process itself will be described in Chapters 7-9.

6.1. Implementation Requirements

The requirements for the implementation of the *Interaction Constraints Model* depend on how the design team that develops the mobile or wearable computer system works and which other computer-aided software engineering tools (CASE tools) it uses. Since this research focuses on the underlying model and its applicability for situation-aware user interface design, I will specifically show the requirements that I had to fulfill to implement a proof-of-concept for the *Interaction Constraint Model*. In a future implementation, these requirements will change according to the software design and development approach for the mobile and wearable computer project.

Most of the requirements for the proof-of-concept can be derived from the objectives for the complete research, since this implementation is intended to prove that the model meets these objectives. In *O-05* and *O-08* (see Chapter 3), I stated the need for an implementation-independent model (i.e., it should be applicable for several possible usage scenarios), and that the model should be readable by humans as well as by machines. The first two requirements for the proof-of-concept implementation are thus as follows:

R-01: The proof-of-concept shall demonstrate at least one implementation method and shall illustrate in which respect other implementations would be better or less applicable.

R-02: The implementation shall demonstrate how a human (a system designer) would use the system (and the underlying model) and it shall illustrate how a computer-supported process would support or replace the system designer's work.

Other requirements result directly from the nature of the proof-of-concept, which should allow for demonstrating the applicability and usefulness of the *Interaction Constraints Model*. The implementation should prove the assumptions of the model, i.e., we can decide about the applicability of user interfaces in analyzing the *constraints* that occur in a specific *work situation*. Furthermore, it should show that domain experts rather than IT experts could use a system that is based on the model. Thus, the next set of requirements can be stated as follows:

R-03: The implementation shall allow for clearly proving the assumptions made for the underlying model (i.e. the possibility of mapping user interfaces to sets of constraints).

R-04: Based on R-03, the implemented system and the data that goes into the implemented system, shall be based on real-world data to enable a sound discussion and meaningful conclusions.

R-05: The implementation shall demonstrate that a system using the Interaction Constraints Model can be intuitive and understandable by domain experts.

I will discuss these requirements and how the implemented proof-of-concept fulfills them in Chapter 9 after discussing possible implementation techniques in the following sections and illustrating the use of the implemented system in Chapter 8.

6.2. Discussion of Implementation Techniques

To implement the *Interaction Constraints Model*, I had to choose from different implementation techniques, i.e. techniques on how to collect and store the *constraints* and user

interface data, and how to view and represent the data. Since view and representation are closely related to data collection and storage, the different techniques can be described based on the actual description language or modeling technique. In the following sections, I describe the four options that I considered: a newly created description language, a UML-based approach, a relational database management system, and an XML-based approach.

6.2.1. Newly created description language

The first category comprises approaches to develop a proprietary description language that would represent the *Interaction Constraint Model* and that would allow for defining model-specific attributes and functions. This new language could be used to describe the *constraints* for different *work situations* and then parse these descriptions to compare the sets of *constraints* to previously entered sets of *constraints* from other projects to see how interfaces performed in these projects, and thus derive advice on finding the right user interface for the right situation. Despite some benefits that a newly developed description language would offer, such as defining an exact fitting taxonomy and algebra, the development of a language for a concept that is not yet proven as useful would be premature. Thus, I see potential for a newly created language only after the proof-of-concept implementation that is goal of this research.

Therefore, I focused on existing implementation options, which include several modeling techniques that are common standards or in the process of becoming standards, such as the Unified Modeling Language (UML), the eXtensible Markup Language (XML), Entity-Relationship Diagrams (ERD) or derivatives of these techniques and languages. The following sections introduce these techniques and my approaches in applying these techniques to the *Interaction Constraints Model*.

6.2.2. UML-based approach

Using the *Interaction Constraint Model* with only the UML would set the focus on a workflow-based analysis and a use-case driven representation, as introduced in Chapter 5 [Kruchten 2000, p. 97] [Jacobson 1992]. This approach would model the interactions between user and mobile or wearable computer in different *Activity Diagrams* or *Use Case Diagrams* and define constraints for certain use cases. In fact, the UML itself defines constraints that can be applied to its diagrams [Fowler 2000, p. 61]. UML even provides an extra language the Object Constraint Language (OCL). The OCL offers a specific notation that allows (mostly) software designers to assign constraints to diagrams defined in the UML notation [Warmer 1998]. Thus, OCL could be used to apply the constraints on the object or class level rather than on the instance level. This means that OCL is a better fit for describing the constraints for groups of objects than for describing specific instances, such as the constraints of *work situations*. Figure 6.1 shows an example of an interface class diagram with constraints for the interfaces' applicability in OCL notation.

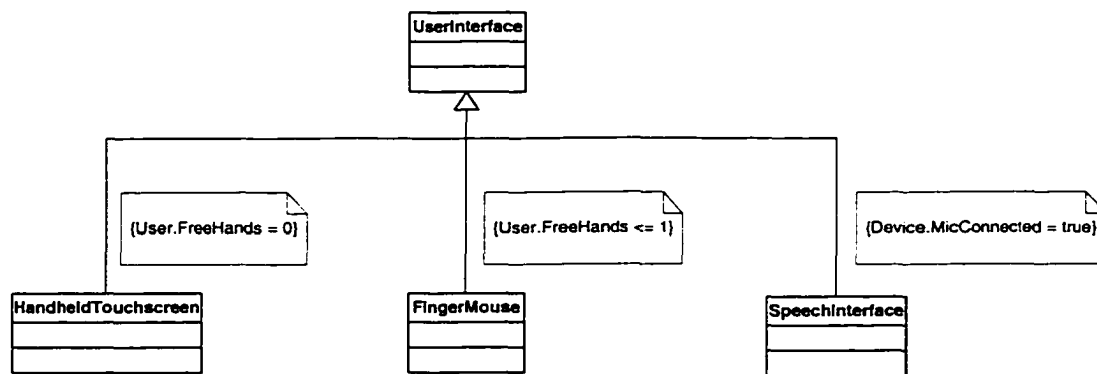


Figure 6.1: Constraints in a UML Class Diagram, constraints are added within brackets; OCL defines the notation of these constraints.

Therefore, besides describing the *Interaction Constraints Model* itself, I use the UML in the task analysis phase as a means of identifying and specifying single *work situations*. In setting

up UML *Activity Diagrams* using swimlanes [Booch 1998, p. 265], we can clearly identify the need for an interface. At each change of swimlanes of the workflow, the system to be developed, either needs a user interface or an interface to another component of the IT infrastructure (see Figure 6.2).

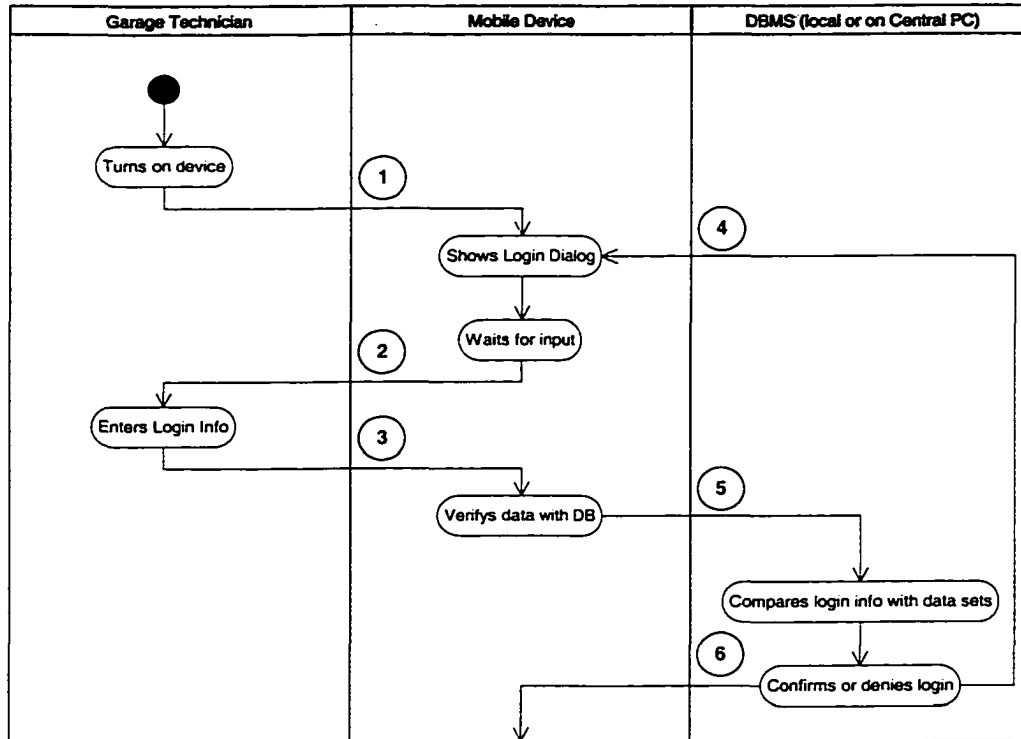


Figure 6.2: UML Activity Diagram using swimlanes. Each change of swimlanes describes either a user interface (#1-3) or an interface to another component of the IT infrastructure, here a database management system, DBMS (#4-6)

6.2.3. Relational Database Management Systems

Entity-Relationship (E-R) Models are capable of describing attributes of objects and their relationships to each other. Relational Database Management Systems (RDBMS) offer even domain experts the possibility of implementing queries over the data sets and of designing dialogs for the data management, such as input and output dialogs. Thus, using an RDBMS to store the data and using the database management system to query the data in the *Interaction Constraints Model* is a conceivable solution. It offers a solid way to store all the *constraints* that

occur in the multiple *work situations* and to link them to the related user interface implementations of the different projects. Through the input and output dialogs, and the available query functionality, even domain experts with limited IT knowledge will be able to use the system. One drawback however, is the limited exchangeability and machine-readability of the data, especially in comparison to an XML-based approach described in the next section.

6.2.4. XML-based approach

The eXtensible Markup Language – XML [Bradley 1998] [Williamson 2001] is a language that describes data in text documents. XML is a markup language similar to HTML, but allows user-defined variables or ‘tags’. With these tags, designers can build up data structures in a kind of declaration document, the Document Type Definition (DTD) or XML Schema Definition [W3C 1997]. This allows a designer to model hierarchical data models with objects and their attributes in these definition files. The actual data files can be verified against these definitions and thus ensure the formal correctness of the contained data. Since XML is based on text documents, it is exchangeable between all sorts of platform and applications. A software application only needs to read and parse the documents to use the data. The XML markup “can be used to describe other languages. This allows you to create your own XML tags that provide information about the information for which you are creating the XML based document structure.” [Williamson 2001, p.9] Thus, there are efforts in different industries to standardize the DTDs and Schemas, many kinds of XML dialects have emerged and are still in progress, such as ebXML for standardized e-business transactions [ebXML 2001], aecXML for XML-based exchange of construction-related data Architecture Engineering and Construction [aecXML 1999], ifcXML for the exchange of data modeled according to the Industry Foundation Classes [ifcXML 2001], VoiceXML for a standard that defines spoken dialogs for automated speech response systems [voiceXML 2000], and the XML User Interface Language which approaches the configuration of user interfaces with an open standard [XUL 2000]. Finally, the RQML [Gudgeirsson 2000] shows an example, where a newly-defined XML dialect was used to describe requirements for software projects. In this

research, Gudgeirsson approached requirements management that is a well-established topic or problem area and therefore his efforts went into a proven concept. For the *Interaction Constraints Model*, as mentioned earlier, it would be premature to put too much effort into creating a new language before proving the concept. However, I see great potential in transforming the model to an XML dialect in the future and to establish a standardized usage of the model. This would well integrate into a visionary concept of distributed XML descriptions of the single components. For example, a bridge could be described in ifcXML, the initial functional requirements for a bridge inspection support system with RQML, an inspection system description with Sunkpho's Java Inspection Framework (JIF) [Sunkpho 2001], and the "instruments" (data collection means) of the JIF or their corresponding *constraints* on the user or speech interaction would be described in the future XML dialect.

6.2.5. Discussion of Implementation Techniques

Based on the insights I got by considering and applying the different techniques to the *Interaction Constraints Model*, I mapped the different techniques to the requirements defined in Section 6.1. Table 6.1 shows this mapping.

	UML	RDBMS	XML
R-01	+	+	+
R-02	-	o	+
R-03	+	+	+
R-04	+	+	+
R-05	o	+	-

Table 6.1: Mapping of the considered implementation techniques to implementation requirements; from Section 6.1 (scale from “-“ over “o” to “+” for “not reached” over “sufficiently reached” to “completely reached”)

Looking at these implementation techniques with respect to the requirements that I defined at the beginning of this chapter, we can see that they only differ for *R-02* and *R-05*, which are the demands that the data of the system should be readable for humans and machines and that the system would be intuitive and easy to use for domain experts with less IT experience.

UML clearly fails in the readability for machines, since computers cannot understand UML diagrams, unless they are part of an integrated CASE tool, which provides use-case modeling with added meta data for each diagram. An example for such a CASE tool is Rational Rose [Rational 2000], [Bormann 2001], which supports the Rational Unified Process [Kruchten 2000], mentioned earlier. These systems however, are themselves based on some kind of database management system, and would thus better fit in the database category. Another drawback of using the *Interaction Constraints Model* with only use cases is the fact that potential users must have a basic knowledge of use cases and that a “query” of the data would mean comparing use cases and use case descriptions.

Using a database management system for the proof-of-concept is the technique I chose because this best supports the readability by humans and the intuitive use for domain experts, which allows me to describe and demonstrate the system more easily. The functionality of a DBMS allows for the creation of input and output masks and the definition of queries with an acceptable effort. Although the readability of databases by machines can be implemented, the readability of the real data and especially is very limited and the data structure is most often proprietary to a specific DBMS.

The XML-based approach in my opinion is the ideal implementation technique, but only after the concept of the *Interaction Constraints Model* is proven. Creating an XML description of the collected data would enable all sorts of access to this data, and even would allow humans to read and edit the raw data. Although the latter suggestion might not be desirable for domain experts, there are still ways of representing and processing the data into an acceptable format. In a visionary scenario, the sets of collected data from previous projects could be stored on a central server and then accessed by web clients through an intranet or the Internet, networking different

design teams using different development environments on different platforms. In an even more visionary step, the automated decision on the right user interface for the right situation could be made on the basis of *constraints* gathered at operation time instead of design time. Furthermore, information needed by the field-engineer could be provided by the different components that are involved in a work situation by individual XML descriptions, such as a user file, a bridge description file or a device configuration file. These files could then be processed to a set of *constraints* for the specific situation and build a decision basis for the right user interface. Thus, the user interaction could become adaptive and context-aware based on the use of the *Interaction Constraints Model*.

6.3. Proof-of-Concept Implementation – *ICE-Tool*

As discussed above, I chose a DBMS to implement the proof-of-concept application of the *Interaction Constraints Model*, called “Interaction Constraints Evaluation Tool” (*ICE-Tool*). I chose Microsoft Access 2000, because it is the most-established DBMS, at least for projects of this scale [Ravey 2002], and because in using a known product, I could build on knowledge I gained during previous projects. *Microsoft Access* offers the functionality I needed to implement the data structure, derived from the model description shown in Chapter 5, and to set up input and output dialogs that enable the system designer to use the application. Finally, I was able to create the necessary queries to sort and display the data and to find similar patterns between different sets of *constraints*. Figure 6.3 shows the Entity-Relationship Diagram (ERD) of *ICE-Tool*, which I derived from the object model that I set up and described in Chapter 5 (see Figure 5.3). It illustrates how the *work activity* and *work location* form a *work situation* and map to the *influences* and *constraints* (right side of Figure 6,3) and to the user interface and project information (left side of Figure 6.3). Thus, the ERD reflects the central position of the *work situation*, which allows users to independently enter and map information about the occurring *constraints* and information about implemented user interfaces to a *work situation*. This allows for accessing the information about implemented user interfaces from the *constraints* side and for accessing information about

constraints that occurred for a specific implemented user interface. This might become clearer when looking at the main screen of *ICE-Tool* (see Figure 6.4). Before the user of the application can enter information in the lower part of the screen, a unique *work situation* has to be defined by selecting a previously entered *work location* and a previously entered *work activity*. If the specific location or activity entry is not yet entered, the user has first to add it to the corresponding list. The lower part of the screen is divided by the list of *constraints* that occur in the specific work situation and contains the information about the causes of the *constraints* (or *influences*) and the implemented user interfaces.

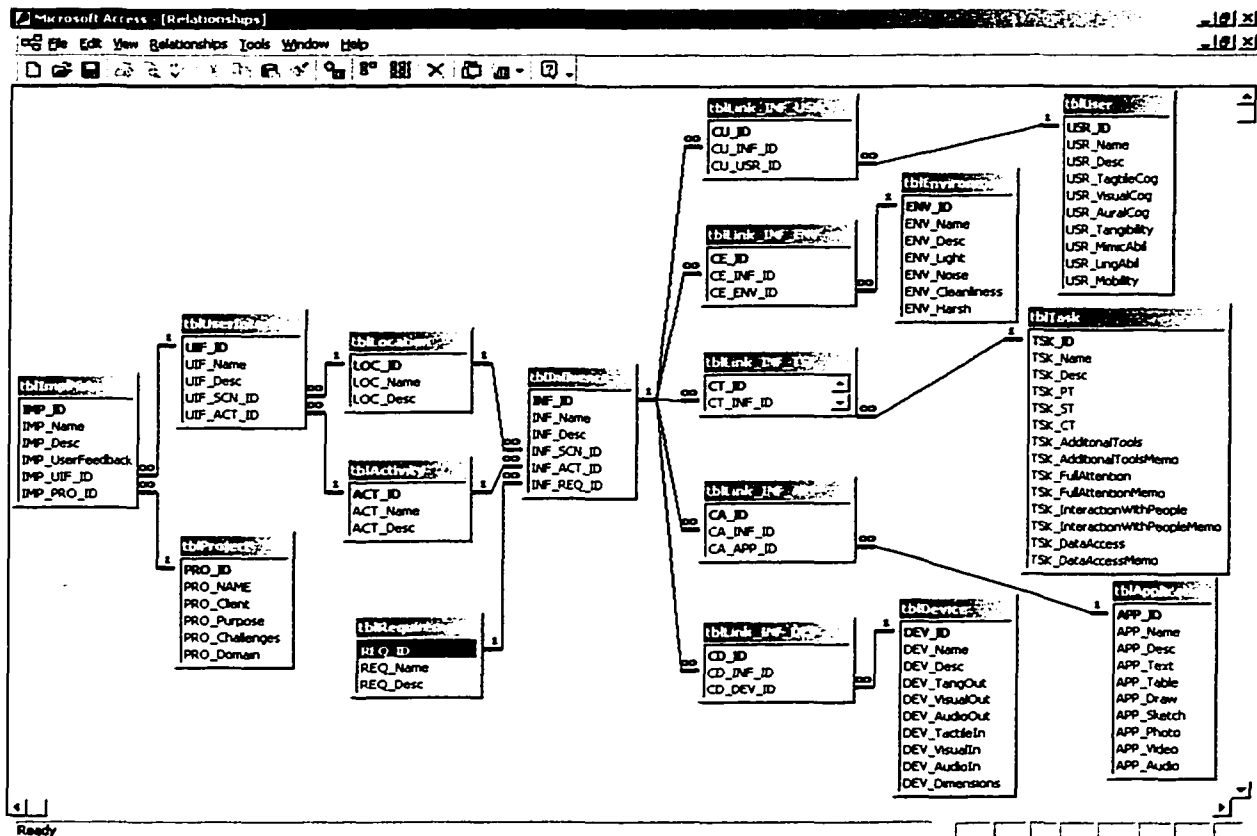


Figure 6.3: ERD of the *Interaction Constraints Model* as implemented in *ICE-Tool*

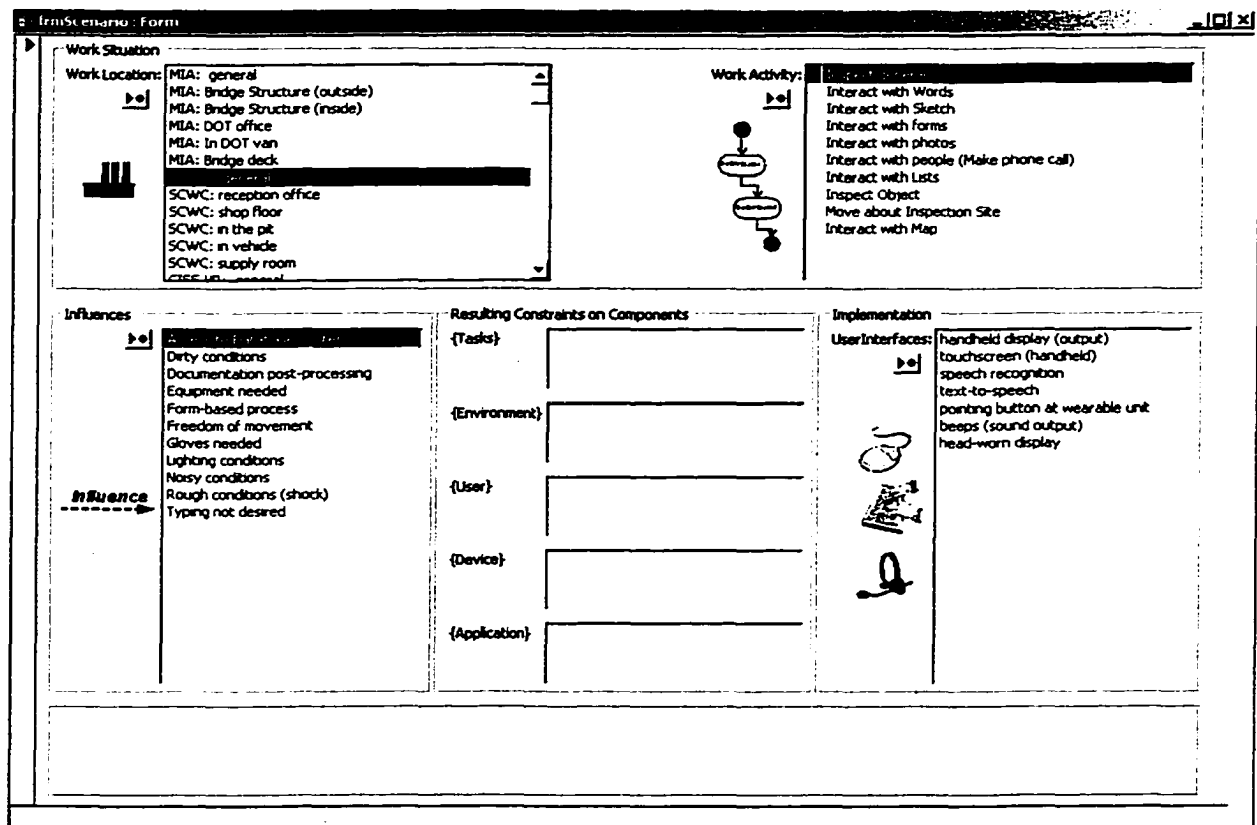


Figure 6.4: Main Screen of the *Interaction Constraints Model* as implemented in *ICE-Tool*

To clarify the relationship of the different list boxes on the main screen to the *Interaction Constraints Model*, each component is indicated by the corresponding icons from the model overview shown in Chapter 5, Figure 5.1.

The list boxes labeled “Resulting Constraints on Components”, shown in the middle of the lower part of the screen and separated into the five *constraint categories* (*Task, Environment, User, Device, and Application*) show the *constraints* that are caused by the influence selected in the list on the lower left part of the screen. To get the impact of the combination of all *influences* of a specific *work situation*, the user can use the sum button. This will query all influences and display the most constraining combination of *constraints* for each of the five components. If for example, a technician has to carry a flash light which blocks one hand and the task needs to be conducted one-handed as well, the sum would be that the User has no free hand, and thus a hands-free user interface would be desirable. Another example would be high ambient noise,

which limits the use of speech technologies, but might be compensated by noise-canceling microphones or a limited command vocabulary that offers fewer sources of errors. However, the complete extent of possible combinations and thus the meaningful query techniques will only become obvious during the use of the application and will be discussed in the illustrative example in Chapter 8.

Based on the sum or set of *constraints (constraint patterns)* for a specific *work situation*, the application allows users to run queries against the data sets of previously entered projects. If the *constraint pattern* matches the set of constraints of previously entered *work situations*, these situations and their related information about user interfaces will be returned as a report. This report summarizes a collection of user feedback about specific user interface implementations that were used for the same *constraint pattern* as those that went into the report.

Figure 6.5 shows an example of such a report. It shows the project name, the domain, the different user interfaces, and some user feedback.

Project	Domain	Interface	Implementation	Location	Activity	Notes
SCWC 1	automotive	head-worn display	XyberView w/ MA IV	SCWC: general	Inspect general	some inspectors said they'd feel like in a star trek movie felt like looking through someone else's glasses
SCWC 1	automotive	handheld display (output)	XyberPanel	SCWC: general	Inspect general	handheld display of Xybermaut MA IV (res. 640x480)

Tuesday, February 26, 2002 Page 1 of 3

Figure 6.5: An *ICE-Tool* report on previously implemented user interfaces.

For this proof-of-concept, this procedure was a good approach for quickly showing the different similar *work situations*. In the future, the query and representation functionality has to be extended. Based on modern data analysis techniques, a detailed search for specific information within the collected data sets can be performed. Of course, to apply data analysis techniques, we first have to establish a solid collection of data. Therefore, the generation of reports seems sufficient for proving the *Interaction Constraints Model* with *ICE-Tool*.

6.4. Evaluation of *ICE-Tool*

To evaluate the proof-of-concept implementation (*ICE-Tool*), I went back to the requirements that I set up in Section 6.1. This evaluation focuses on the actual implementation and not on the usefulness of the whole system, which will be discussed in Chapters 8 and 9, where I will show a complete walk-through of *ICE-Tool* using some real-world data that I collected and entered as described in Chapter 7.

The *ICE-Tool* implementation meets the five requirements as follows:

The implementation fulfils *R-01*, the demonstration of the possibility to implement the *Interaction Constraints Model* in offering the complete functionality as shown in the previous section. The advantage compared to a UML-based or only use case-driven approach is demonstrated by the available data queries and representation possibilities. A disadvantage of the RDBMS is the completely table-based representation, which requires some abstraction capability to visualize the relationships between the single components, such as *work location*, *work activity* and the *influences*. Compared to an XML-based implementation, the system was easier and faster to implement, which is important for a proof-of-concept. Although the XML approach would offer more flexibility, the database implementation of *ICE-Tool* shows enough functionality to appropriately discuss the value of the *Interaction Constraints Model*.

The second requirement, *R-02*, asked for the demonstration of how a human uses the system and an illustration of an automated process. The previous section showed the main screen and a report of *ICE-Tool*, Chapter 8 will present a complete walk-through of the system and illustrates its usability. An automated process would measure and sense the occurring *constraints* in each situation and then access the database. Based on similar previous situations, and based on the amount of automated data analysis that is built in, the system will decide which user interfaces would be applicable in the specific situation and additionally it could include the user preferences that the system collected during previous usage by this user.

Requirement *R-03*, demands the proof of the assumptions that user interfaces can be mapped to *constraints*. The DBMS shows the possibility to create a data structure to capture the needed data and shows how a user can access the result of data queries on this data source. This does not mean that *the Interaction Constraints Model* and the process of choosing the best interfaces for a situation is proven positively, but it shows that *ICE-Tool* can be used to prove the model.

R-04, as stated above, is based on *R-03*, and thus, its fulfillment is also related to *R-03*. However, at this point, only the possibility to store real-world data is proven. So far, this implementation is an “empty” database that will be filled with real-world data derived from projects, which I describe in Chapter 7.

For meeting *R-05*, I chose a DBMS, MS Access, which is widely known and offers the user interfaces from the Microsoft Windows operating system. Thus, I built a dialog-based system that wraps any direct manipulation of the data source. Furthermore, I arranged the user interfaces in a way that is natural to the data collection and evaluation process and that allows for approaching the *constraints* data from the influence part as well as from the user interface part. Furthermore, it does not matter which kind of information (*constraints* or user interfaces) is entered at which time, as long as it is mapped to the corresponding *work situation*, i.e. *work location* and *work activity* combination.

7. Example Data from Real-World Projects

To evaluate the *Interaction Constraints Model*, the proof-of-concept implementation prototype had to be populated with example data to allow for an analysis of the concept. In entering data from projects in which I personally participated as well as from projects from colleagues and from the literature, the model could be evaluated based on real-world data and with a broad base of different applications and implementations. This chapter first gives an overview of the different projects and then describes the different projects in separate sections, which identify the *work situations* that I used for my analysis.

7.1. Description of Investigated Projects

For proving the concept with the implementation prototype, I wanted to have a data set that is sufficient enough to find *constraint patterns* that can illustrate the concept of the *Interaction Constraints Model*. I extracted data from research projects in which I was involved in and from projects of colleagues and of other research teams that I found in the literature. The following are summaries of the projects. At the end of each section, I summarized the *constraints* that I included in my evaluation process, separated into their specific *work locations* and *work activities*. Each of the following sections covers one particular project that ends with a short discussion of special issues concerning user interaction for the specific application.

The projects can be grouped in two ways: 1) according to the affiliation of the research team and 2) according to the domain for which the project was conducted. The order of following chapters reflects the affiliation, whereas Figure 7.1 shows both, the affiliation and the domain. The affiliations are separated into projects from the *m/w-CAE Systems Lab* within the *Department of Civil and Environmental Engineering* at *Carnegie Mellon University*, projects from other departments at *Carnegie Mellon University*, and projects from other research groups. The domains are divided into *construction & infrastructure monitoring*, *maintenance & manufacturing*, *automotive*, and *non-industrial*. The last category served as proof for the domain-independence

and showed that although the concept of the *Interaction Constraints Model* focused on industrial applications, it can be applied to a certain degree to non-industrial applications as well.

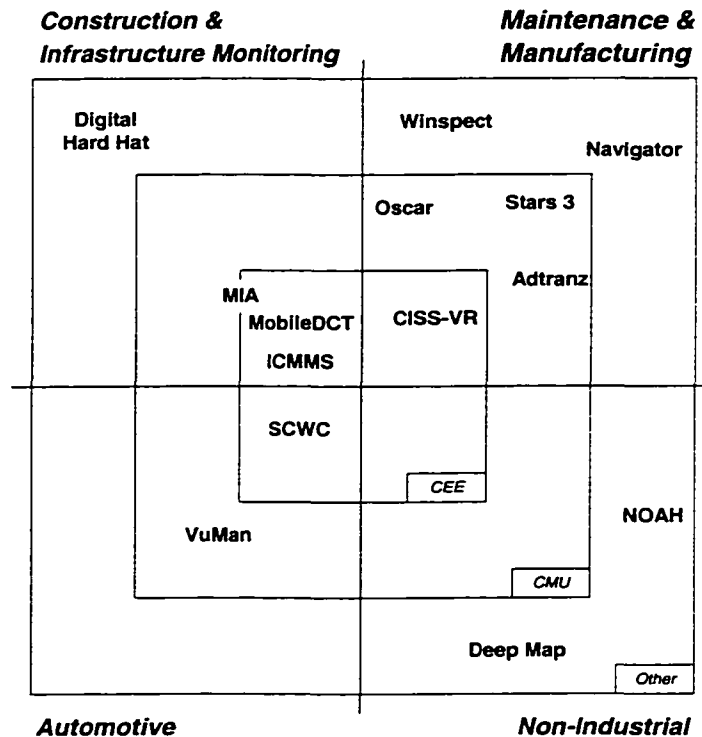


Figure 7.1: Overview of projects divided by affiliation and research groups

First, I will introduce the projects in which I was personally involved (Sections 7.2 and 7.3), and proceed with other projects of the department of *from the m/w-CAE Systems Lab* (Sections 7.4 - 7.6), which are followed by projects from other departments at *Carnegie Mellon University* (Section 7.7 – 7.11), before I illustrate the projects that I found in literature (Sections 7.12 – 7.15).

The project summaries start with general facts and a description with some images of each project to get a general idea of the diversity of the projects before describing the different components that compose the *work situation* identified in each project, listed in separate tables.

7.2. SCWC – Vehicle Inspections

Project:	“Speech-Controlled Wearable Computer (SCWC) for Vehicle Inspections”
Research Team:	<ul style="list-style-type: none">• Carnegie Mellon University, Department of Civil and Environmental Engineering, Pittsburgh, PA, USA• Robert Bosch Corporation, Research and Technology Center, Pittsburgh, PA USA• Robert Bosch GmbH, Plochingen, Germany
Domain:	Automotive
Purpose:	Support garage technicians during vehicle inspection on the shop floor
Challenges:	Rough environment; oily conditions / lack of cleanliness, no mobile or wearable device applied in this environment before.
Device:	Xybernaut Mobile Assistant IV Proprietary DECT-based Bosch device
References:	[Buergy 2000], [Buergy 2001], [m/w-CAE Systems 2002]

Description:

The objectives of this project were to develop a prototype of a hardware and software system that could be presented to and actually used by garage technicians, helping the project team to evaluate the needs and acceptance of future commercial systems. Therefore we took two different approaches with two different devices. One device is based on a commercially available wearable computer – the Mobile Assistant IV (MA IV) from Xybernaut Corporation that transmits inspection data at the beginning and the end of each inspection. The second device is a proprietary hardware system developed during this project, in which a remote server provides the processing power and the memory necessary to run the actual inspection application software. This server can simultaneously handle up to four clients and could be any standard desktop computer available in the repair shop, even the computer at the main office. The idea was to

investigate the relative acceptance of two different systems by service technicians of two approaches: 1) a bigger, more complete system; and 2) a smaller system that provides the same information in a more resource-efficient way (with respect to size, weight and costs). This would enable us to propose a device that can be used for future inspection-related tasks in the automotive industry and similar domains.



Figure 7.2: Garage technicians with paper-based process (left) and using the SCWC 1 with the head-mounted display (middle, right)

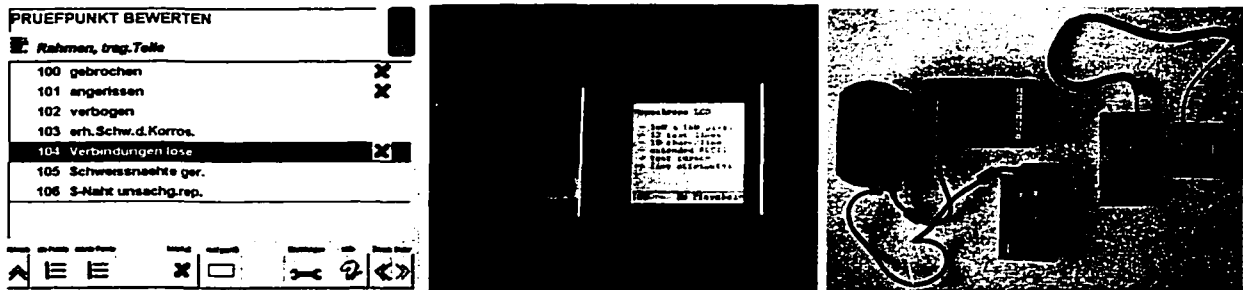


Figure 7.3: (left to right) SCWC 1 graphical user interface, text-based LCD display of SCWC 2, hardware comparison of SCWC 1 (left in third image) and SCWC 2

During the design of the two systems, we interviewed the targeted users, which are real service technicians at repair shops actually performing the inspections without IT support and supported by the first, self-contained prototype of this device, the *SCWC 1*. This first prototype was based on commercial off-the-shelf (COTS) hardware and enabled us to perform field tests before the actual implementation of the final design of the second prototype, the *SCWC 2*. Thus,

we got valuable insights into the requirements for the hardware and software design crucial for the success of the next version of this product.

The functionality provided by this system is completely speech-enabled and thus able to be operated hands-free. This functionality includes: access to centrally entered vehicle and order data; hands-free data collection in the garage environment; remote control and access of measurement devices; communication with other service technicians or repair shop personnel; remote data access; or remote automation control via modem. These functionalities are all directly usable from the mobile unit.

The conditions, in which the devices are intended to operate, are typical shop floor conditions, with technicians who have oily hands, a high ambient noise due to running vehicle engines, and bad lighting in and around the pit. The technicians are familiar with using the central computer that stands near the pit, but did not use mobile or wearable computers during their work.

Identified Work Locations:

SCWC Locations		
1	Reception Office	In the office where a secretary checks in the vehicles
2	Shop Floor	On the shop floor, around the vehicle
3	Pit	In the narrow pit, mostly below the vehicle
4	Vehicle	During test drive or to move the vehicle onto the pit, the technician has to get into the vehicle
5	Supply Room	In the room where the technician gets spare parts, tools, etc.

Table 7.1: Location examples for SCWC project

Identified Work Activities:

SCWC Activities		
1	Interact with Form	The technician has to retrieve, fill out, and submit the form for the right vehicle.
2	Interact with List	The technician retrieves and scrolls inspection lists and checks faulty items.
3	Interact with People	The technician communicates with the main office or colleagues.
4	Inspect Object	The technician uses a tool and / or a flashlight for inspecting detail of the vehicle.
5	Move about Inspection Site	The technician walks around the vehicle, or between the pit and the supply room or main office.

Table 7.2: Activity examples for SCWC project

Implemented User Interfaces:

SCWC Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Beeeps (binary sound output)	Output
7	Pointing Device at Wearable	Input
8	LCD (busy signal of device)	Output

Table 7.3: Implemented user interfaces for SCWC project

Discussion of used interfaces:

In this project, the main issues was to provide the list of 300 inspection items to the garage technicians in a way so they could choose from this list and mark any faulty items. Audio output by the system could only insufficiently be implemented in this system, because of the huge amount of list items that would have to be read out by the system. After discussions with the technicians, we decided to definitely include a graphical output either as a handheld or as a head-worn display. The handheld display did not meet the demands for hands-free operation of the device and the head-worn display was too distracting and too obtrusive during the inspection. Thus, we designed and implemented a text-based display that could be worn on the wrist like a watch and provided only the essential information – the inspection items. A speech recognition engine running on the server provided a means to enter the inspection results. The garage technician preferred the smaller, text-based version of the display.

7.3. CISS-VR – End-of-Line Inspection in Manufacturing

Project:	“Computer Integrated Sight-check System (CISS)”; using speech recognition and Virtual Reality 3D Objects
Research Team:	<ul style="list-style-type: none">• Carnegie Mellon University, Department of Civil and Environmental Engineering, Pittsburgh, PA, USA• Robert Bosch Corporation, Research and Technology Center, Pittsburgh, PA USA• Robert Bosch GmbH, Schwieberdingen, Germany
Domain:	Manufacturing
Purpose:	Support sight-check / visual inspection for end-of-line inspection and assembling processes
Challenges:	Rough environment; oily conditions / lack of cleanliness, visualization of inspection object and localization of inspection items has to be easy.
Device:	Xybernaut Mobile Assistant IV Several pen tablet PCs: Siemens Stylistic, Logic Instrument Tetralight
References:	[Buegy 2000a], [m/w-CAE Systems 2002]

Description:

In the CISS-VR project, we created another prototype for use in manufacturing end-of-line inspections. This system is able to provide a Virtual Reality 3D image of the product that is to be inspected. The GUI shows the specific inspection items, zooms in to them and highlights the problem area. It also provides a short text message on the problem. This prototype can be operated completely by speech and thus without losing the hands for the actual inspection task.

The motivation for this project was to increase the accuracy and efficiency of visual inspection where different products with ever changing inspection items need to be localized. The system helps – based on real images that for a virtual 3D image – to illustrate the actual fault of the item in the as-built status, which would not be possible with ‘clean’ CAD images.

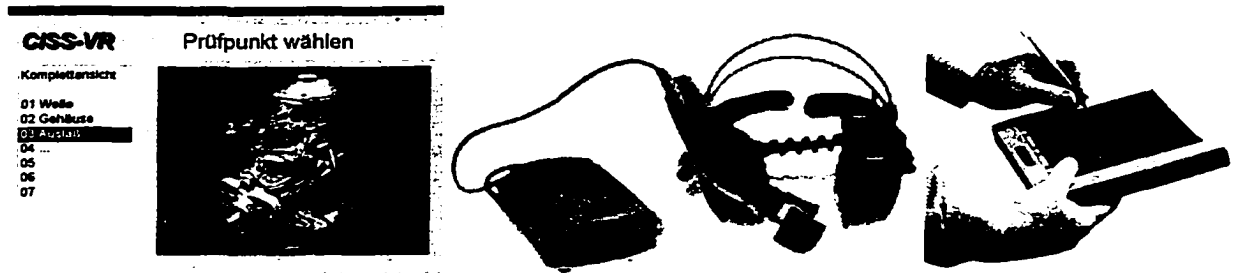


Figure 7.4: (left to right) CISS-VR graphical user interface, Xybernaut MA IV, Fujitsu Stylistic

This prototype was a first step into an Augmented Reality (AR) equivalent that would automatically recognize the inspection objects and adds, or augments, the real-world images with comments and arrows to highlight relevant inspection items. This AR-based system can in future support worker during an assembly job with the guidance that currently is provided by videos shown close to the workplace or by user manuals on the workplace's desk.

Identified Work Locations:

CISS-VR Locations		
1	Assembly Seat	During assembly at conveyor belt
2	Sight-check Point	At the actual visual inspection seat
3	Faulty-part Shelf	At shelf or cart, where the faulty parts are sorted out
4	Shop Floor	Supervisor or inspector walking around checking items on carts and random checks on conveyor belt
5	In Office	In manager's or central office where blueprints etc, are located

Table 7.4: Location examples for CISS-VR project

Identified Work Activities:

CISS-VR Activities		
1	Interact with List	The inspector / assembler retrieves and scrolls inspection lists and checks faulty items.
2	Interact with Photo	The inspector / assembler checks inspection objects against real pictures of faulty parts.
3	Interact with Sketch	The inspector / assembler checks measurements against sketches and blueprints.
4	Inspect Object	The inspector / assembler uses a screwdriver or wrench for inspecting detail of the vehicle.
5	Move about Inspection Site	The inspector / supervisor walks around checking items on carts and on conveyor belt.

Table 7.5: Activity examples for CISS-VR project**Implemented User Interfaces:**

CISS-VR Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Pointing Device at Wearable	Input

Table 7.6: Implemented user interfaces for CISS-VR project**Discussion of used interfaces:**

The special issue of user interaction in the CISS-VR project was the opportunity to control a 3D-image of an inspection object with using speech recognition and thus hands-free. This provided a useful means of identifying special features such as inspection items more precisely and faster. Since the system could also be operated with the graphical user interface, it also was applicable for use on pen tablet computers and other handheld touchscreens.

7.4. ICMMS – Construction Progress Monitoring

Project:	“Integrated Contract Management and Monitoring System (ICMMS)”, supporting construction progress management
Research Team:	<ul style="list-style-type: none">• Department of Civil and Environmental Engineering at Carnegie Mellon University, Pittsburgh, PA, USA.• Technical University of Dresden, Germany• HOCHTIEF, Germany
Domain:	Construction
Purpose:	To reduce the paperwork and redundant data entry during construction progress monitoring.
Challenges:	Rough environment; windy, wet conditions / lack of cleanliness, inspection locations elevated above ground, climbing often necessary, access to previous reports and blueprints desired.
Device:	Xybernaut Mobile Assistant IV Several pen tablet PCs: Siemens Stylistic, Walkabout
References:	[Reinhardt 2000], [Garrett 2002], [m/w-CAE Systems 2002]

Description

The ICMMS project had the goal to reduce paperwork and especially to eliminate the multiple data entry from paper-based data collection of progress monitoring. Transferring drawings and sketches of the construction process in particular, could be speeded up because the process of marking up the progress in a drawing and calculating the progress based on these drawing became obsolete by using the system. With an easy-to-use graphical interface, the system provided a means of entering the progress with a few strokes on the touchscreen. In entering the progress data right at the construction site the system could provide immediate feedback on the element's progress as well as on the overall progress. Through this feedback,

the progress engineer gets a good idea of the importance of the single elements for the completion of the overall project.

The system also provides means of adding comments about specific activities or elements to capture information that exceed the actual progress information or that help to clarify some of the entered data. The system also offers speech recognition and speech synthesis to enable audio interaction with the device.



Figure 7.5: (from left to right) ICMMS graphical user interface, worker using the system, Xybernaut MA IV running ICMMS

ICMMS is a prototype of a Progress Monitoring System. It has been designed to enhance the efficiency of progress monitoring for construction processes that, comprises the set up of the monitoring system as well as the collection, processing and evaluation of progress data. Tests of ICMMS on a real project showed that although the setup of the initial monitoring data is very time consuming, the overall process reduces the data collection time to approximately a third of the paper-based process. Even external or not as well skilled personnel can perform weekly updates of the construction process in near-real time. Through this process the construction progress becomes more transparent and more controllable.

Identified Work Locations:

ICMMS Locations		
1	Construction site office	The construction manger's on-site office
2	Finished building floor	On a floor that is finished and cleaned up
3	Scaffolding	On scaffolding somewhere on the construction that is still in progress
4	Excavation Area	In an area, where excavation is going on and where the surface not yet leveled
5	Tunnel / Basement	In a tunnel construction or in a basement of a building construction.

Table 7.7: Example locations of ICMMS project

Identified Work Activities:

ICMMS Activities		
1	Interact with List	The construction manager has to choose the structure components that are to be monitored / evaluated.
2	Interact with Photo	Sometimes picture proofs are required.
3	Interact with Map	To retrieve the actual location of structural components, they must be localized on a map..
4	Inspect Object	The construction manager has to check / verify components using a tool or a flashlight.
5	Move about Inspection Site	The inspector has to walk / climb the site and check the structure at different locations.

Table 7.8: Example activities of ICMMS project

Implemented User Interfaces:

ICMMS Interfaces		
	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Pointing Device at Wearable	Input

Table 7.9: Implemented user interfaces for ICMMS project

Discussion of used interfaces:

The special interface in this project is the easy-to-use graphical interface that allowed for entering progress data with just a few clicks on the touchscreen. Since construction sites tend to be noisy environments that sometimes disallow the use of speech recognition, the progress managers in the field tests preferred this option of data entry. Furthermore, progress managers are used to carry a clipboard for the paper-based process and thus might not mind having their hands busy carrying a pen tablet computer.

7.5. MobileDCT – Landfill Monitoring

Project:	“Mobile Data Collection Tool (Mobile DCT)”
Research Team:	<ul style="list-style-type: none">Iris Meissner, graduate student at the Institute for Numeric Methods and Informatics in Civil Engineering at Technical University Darmstadt, Germany as visiting researcher at the Department of Civil and Environmental Engineering at Carnegie Mellon University, Pittsburgh, PA, USA
Domain:	Construction
Purpose:	Support landfill monitoring processes with a computer-supported data collection tool, which helps to reduce paperwork and multiple manual data entry and handling.
Challenges:	Rough environment; windy, wet, access to previous reports desired, gloves and sometimes facemasks required.
Device:	Xybernaut MA IV and Magellan GPS.
References:	[Meissner 2001], [Meissner 2002], [m/w-CAE Systems 2002]

Description:

The MobileDCT project was aimed to develop a wearable computer system that could cover the field data collection for a landfill monitoring system. This system was developed within the framework of a research project at the Institute for Numerical Methods and Informatics in Civil Engineering at Technical University in Darmstadt, Germany. The landfill monitoring system developed in this research is a practical system to manage and document the structure, the business management and the administrative information of landfills during their entire lifecycle.

Federal regulations force managers of landfills to take different measures to prevent pollution (gases, noise, smells etc.) above the surface or ground waters below the surface as well as effects on neighboring property. The impact of land filling on the environment and on public health has to be minimized. Therefore, regular measurements have to be performed and the

emissions through the cover of the landfill have to be documented. Thus, on-site inspections are necessary to assess a location, to supervise construction and operation, to monitor emissions and to assess the effect of rain or other influences on the processes inside.

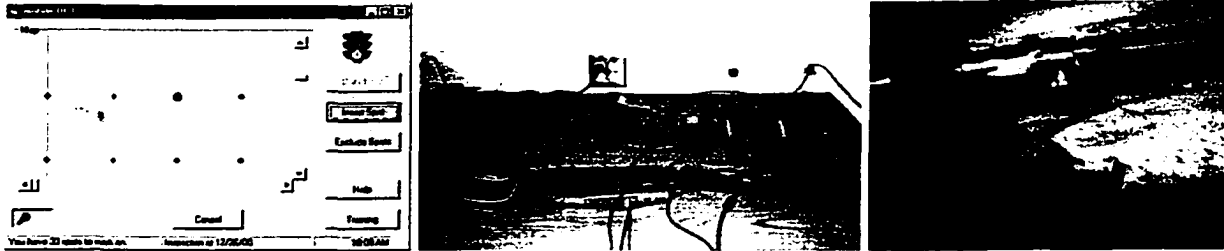


Figure 7.6: (from left to right) MobileDCT graphical user interface, Xybernaut MA IV with Magellan GPS, system in use

Without mobile IT support and especially without the aid of a Global Positioning System (GPS), the landfill inspector collects and documents data on paper forms with an estimated position of the emissions readings. With the use of GPS data, the inspector can document the locations of the measurements more accurately and can retrieve those during the next on-site inspection. Furthermore, the system helps the inspector to find specific reading locations and guides visually (with a map) and acoustically (with series of beeps) to the final reading location.

The MobileDCT consists of a Xybernaut Mobile Assistant IV, a connected Magellan GPS receiver and the speech-enabled application software that provides an interface to the stationary landfill monitoring system. The landfill inspector carries a gas detector, which is necessary for taking the measurements and thus occupies at least one hand. During the measurements both hands are occupied. Thus, a hands-free interface is required for an uninterrupted workflow.

Identified Work Locations:

MobileDCT Locations		
1	Landfill Office	The office of the landfill manager / landfill reception.
2	Landfill Vehicle	In the vehicle that helps the inspector to get around the landfill
3	Covered Landfill	On the portion of the landfill that is already covered and easily accessible
4	Uncovered Landfill	On the portion of the landfill that is not yet covered and hard to move about caused by landfill debris
5	Reading Spot	At the actual reading spot (near a ventilation outlet)

Table 7.10: Example locations of MobileDCT project

Identified Work Activities:

MobileDCT Activities		
1	Interact with List	The landfill inspector has to browse the list of inspection instruments / reading spots.
2	Interact with Map	The inspector checks position on the GPS-supported map.
3	Inspect Object	The inspector has to measure emissions at specific locations.
4	Move about Inspection Site	The inspector has to walk about the site and sometimes crawl through the debris.
5	Interact with People	The inspector communicates with the landfill manager or drivers of landfill trucks.

Table 7.11: Example activities of MobileDCT project

Implemented User Interfaces:

MobileDCT Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Pointing Device at Wearable	Input

Table 7.12: Implemented user interfaces for MobileDCT project**Discussion of used interfaces:**

In the MobileDCT project, the wearable computer system provided feedback about the user's position in two different ways: it showed the position on a simplified map of the site and simultaneously gave acoustic feedback, which indicated the proximity to the measurement location. This audio feedback proved very useful since the inspectors usually know the route they walk for a series of measurements. Thus, the inspector could walk in the approximate direction of the next measurement location and received feedback if they reached their destination. In this way, the inspector had to take out the handheld screen only occasionally. This project reflects this idea in another implementation that used the Vocollect system [Vocollect 2001], which is an audio-only wearable computer system that is used primarily in warehouses for commissioning.

7.6. MIA – Bridge Inspections

Project:	“Mobile Inspection Assistant (MIA) supporting bridge inspections”
Research Team:	<ul style="list-style-type: none">• Student Research team of course “Rapid Prototyping of Computer Systems” at Carnegie Mellon University, Institute for Complex Engineered Systems, Pittsburgh, PA, USA• <i>Jirapon Sunkpho</i>, doctoral student at Department of Civil and Environmental Engineering at Carnegie Mellon University, Pittsburgh, PA, USA
Domain:	Infrastructure Monitoring
Purpose:	Support bridge inspectors in their bi-annual inspection of bridges, providing previous and current data about the infrastructure and especially reduce the paperwork and redundant data entry.
Challenges:	Rough environment; windy, wet conditions / lack of cleanliness, inspection locations elevated above ground, climbing often necessary, access to previous reports and blueprints desired.
Device:	Carnegie Mellon's TIA P wearable computer. Via II, wearable computer
References:	[Balasubramanian 1998], [Garrett 1998], [Garrett 2000], [Sunkpho 2000], [Sunkpho 2001], [m/w-CAE Systems 2002]

Description:

The purpose of the Mobile Inspection Assistant (MIA) is to reduce the amount of paperwork and redundant data entry necessary for filing inspection reports, while providing more convenient access to previous inspection reports. The main goals of the project included minimization of the amount of time spent on paperwork, so that the inspectors can spend more time on the actual inspection, and providing a user interface that can be easily used and providing near hands-free operation. Bridge inspectors tested laptop computers with desktop-oriented software in the field before, but did not approve their use. The inter-disciplinary team from Carnegie Mellon University, researched and designed an integrated system of hardware and

software. This system provides inspector-friendly support for field data collection and decision-making and thus recognizes the field-oriented nature of bridge inspection. The result of this project was a first prototype of a wearable computer system that supports bridge inspectors. The group field-tested this prototype and received feedback from bridge inspectors of the Pennsylvania Department Of Transportation (PennDOT) in Pittsburgh, PA, USA.



Figure 7.7: (left to right) Bridge inspector in paper-based and wearable computer-supported process, CMU's TIA P wearable computer.

The team interviewed bridge inspectors, attended a field inspection, and a seminar on bridge inspection methods. Based on the requirements derived from talking with and observing bridge inspectors, the team created a usage scenario of the envisioned product. After that, they developed a design scenario by systematically walking through every step involved in using the product. Hereby, the team identified and resolved several issues with respect to design and implementation. The group then designed and developed the different components of the system in individual teams and combined them later to a working system. The group regularly obtained feedback from the bridge inspectors at the end of each design phase, to ensure that they would meet the design goals.

Identified Work Locations:

MIA Locations		
1	Bridge Structure (outside)	On the outside bridge structure
2	Bridge Structure (inside)	On the inside bridge structure
3	Bridge Deck	On top of a bridge
4	DOT Office	In the DOT's office preparing or post-processing data
5	DOT Van	In the DOT's vehicle parking close to the bridge

Table 7.13: Example locations of MIA project**Identified Work Activities:**

MIA Activities		
1	Interact with List	The Bridge Inspector has to choose, which bridge to inspect and which bridge components to evaluate.
2	Interact with Photo	The inspector has to document some of the damages with photographic images.
3	Interact with Sketch	To describe the location of damages in detail, inspectors prefer to draw sketches.
4	Inspect Object	The inspector has to measure the thickness of cracks, scrub off rust, and carry some tools to inspect the components.
5	Move about Inspection Site	The inspector has to climb the bridge and check the structure at different locations.

Table 7.14: Example activities of MIA project

Implemented User Interfaces:

MIA Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Pointing Device at Wearable	Input

Table 7.15: Implemented user interfaces for MIA project**Discussion of used interfaces:**

For this system, the main issue of the user interaction was to provide all data the bridge inspectors need for performing an efficient inspection. Since this data contained graphical information, such as sketches and blueprints, the system has to provide some kind of display. In an early prototype the system had a display hanging in front of the inspector, which was not approved by the inspectors. Another factor was that the system requirements asked for a very rugged design and even protection against a possible drop into water.

7.7. Navigator – Aircraft Assembly

Project:	“Boeing’s Wire Bundle Assembly Project” supporting the assembly and montage of cables in aircrafts with the Navigator wearable computer.
Research Team:	<ul style="list-style-type: none">• Institute of Complex Engineered Systems at Carnegie Mellon University• Boeing Computer Services
Domain:	Manufacturing
Purpose:	The assembly of cabling in aircrafts cannot be automated and thus has to be done by technicians. Using mobile IT support enhances and speeds up this assembly process.
Challenges:	Technicians have to crawl within the aircraft and thus cannot carry too much weight or obtrusive hardware. The assembly descriptions are contained in paper manuals. The information within these manuals can change as fast as on a weekly basis.
Device:	CMU’s Navigator.
References:	[CMU 1996], [Smailagic 1999], [Bass 2001] [Wearlab 2002], [MTO 2002], [MTO 2002a]

Description:

The assembly of aircrafts and especially the mounting of cables is a task that cannot be completely automated. Thus, the approach of this research was not to eliminate humans in the assembly line, but to enhance their capabilities by a wearable computer. This computer was intended to provide the ever-changing assembly manuals in electronic form with providing the most current updates. Without the computer-supported process, the cabling blueprints were posted on long posters along the assembly shop floor and thus not easily accessible during the actual assembly activity. The computer system indicated the location of cable fixtures to the technician by a head-worn display, and thus provided real-time help at the actual task.



Figure 7.8: (from left to right) Boeing's assembly shop, Navigator 2, system in use

The system contained the complete documentation of the aircraft, facilitated the inspection and troubleshooting of the assembled components with electronic checklists and fault detection help. The system tracked the user's head and illustrated the location of the cabling in the head-mounted display. Thus, the user did not have to turn eyes away from the actual task. In training sessions, users of the wearable computer were able to get step-by-step description of "best practices", and thus to learn the ideal assembly process. The hardware first consisted of commercial-of-the-shelf components and was later substituted with a system designed by Carnegie Mellon University, the Navigator 2.

Identified Work Locations:

Navigator Locations		
1	Shop Floor	Technician walks between stations on the shop floor of the maintenance hall.
2	Aircraft (inside)	Within aircraft, to be assembled / inspected.
3	Scaffolding (outside aircraft)	On scaffolding supporting jobs performed on the aircrafts shell.
4	Aircraft (below, in maintenance hall)	Below the aircraft, in the maintenance hall.
5	Aircraft (outside, outdoors)	Around the aircraft at an outdoor parking spot.

Table 7.16: Example locations of Navigator project

Identified Work Activities:

Navigator Activities		
1	Interact with List	Aircraft technician retrieves assembling tasks.
2	Interact with Sketch	Technician looks up sketches during assembly or draws sketches during maintenance.
3	Inspect Object	The technician assembles or inspects a component of the aircraft.
4	Move about Inspection Site	Technician moves between different stations such as aircraft, supply room, etc.
5	Interact with People	Technician asks expert at hotline.

Table 7.17: Example activities of Navigator project**Implemented User Interfaces:**

Navigator Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output

Table 7.18: Implemented user interfaces for Navigator project**Discussion of used interfaces:**

The special user interaction used in this project was the combination between speech input and the use of a pointing device. The motivation for this concept was the concept of the graphical user interface that showed a figure of the aircraft, which was divided into several small sections. If the inspector wanted to add a comment about a specific item, the system was waiting for a selection of an aircraft section by the pointing device followed by a speech command. The vocabulary for a specific time of the inspection was adapted to the selection on the graphical interface. This concept saved the inspectors from navigating the section by speech, which can be a tedious job if the displayed sections are small and thus of a high number of selection options.

7.8. VuMan – Vehicle Inspection

Project:	“VuMan”, supporting U.S. Marines during vehicle inspections
Research Team:	<ul style="list-style-type: none">• Institute of Complex Engineered Systems at Carnegie Mellon University
Domain:	Automotive
Purpose:	Assist U.S. Marines in performing a Limited Technical Inspection (LTI)
Challenges:	Shop floor and outdoor inspection locations, inspectors need hands for inspection
Device:	CMU’s VuMan 3.
References:	[CMU 1997], [Smailagic 1999], [Bass 2001] [MTO 2002], [MTO 2002a]

Description:

The VuMan project provided mobile IT support for vehicle inspections. The system allowed to navigate through a hierarchy of inspection items, using an input mechanism that was designed during this project: a rotary dial, which was usable even with gloves was attached to the body of the wearable computer and allowed the selection of specific menu items displayed on the head-mounted display. The inspectors could perform the data collection during the inspection by using only one hand, which kept the other hand free for inspection tasks or for carrying a tool.

The design of the system was co-design of electronics, mechanical and software designers. With this complete hardware and software design, the interaction with the device could be optimized with respect to the integration of input and output interfaces.



Figure 7.9: (from left to right) VuMan graphical user interface and VuMan system in use

Identified Work Locations:

VuMan Locations		
1	Shop Floor	On the shop floor, around the vehicle
2	Pit	In the narrow pit, mostly below the vehicle
3	Vehicle (outside, outdoors)	On an outside inspection site or in the field, mostly below the vehicle

Table 7.19: Example locations of VuMan project

Identified Work Activities:

VuMan Activities		
1	Interact with Form	The technician has to retrieve, fill out, and submit the form for the right vehicle.
2	Inspect Object	The technician uses a tool and / or a flashlight for inspecting detail of the vehicle.
3	Move about Inspection Site	The technician walks around the vehicle, or between the pit and the supply room or main office.

Table 7.20: Example activities of VuMan project

Implemented User Interfaces:

VuMan Interfaces		
1	Rotary Dial	Input
2	Head-mounted Display	Output
3	Speech Recognition	Input

Table 7.21: Implemented user interfaces for VuMan project**Discussion of used interfaces:**

The VuMan's key feature is a rotary dial input user interface, which enables a one-handed input mode. Therefore, also the software was designed in a rotary orientation and the user can turn the dial to a specific item and click to select, even with heavy working gloves. The system can be used more easily than other pointing devices for mobile use because it offers only one degree of freedom. This fact also allows for less expensive manufacturing costs because the mechanical system is simpler. With the rotary dial, this research team introduced "a new interface paradigm: circular input, circular visualization" [Smailagic 1999]. To increase the interaction possibilities with the dial, the system also offered different buttons at the device that could be pressed simultaneously with the dial movement and thus triggered different events.

7.9. Adtranz – Maintenance and Collaboration Tool

Project:	“Adtranz Mobile Computer: supporting maintenance and online collaboration for maintenance
Research Team:	<ul style="list-style-type: none">• Institute of Complex Engineered Systems at Carnegie Mellon University, Pittsburgh, PA, USA• Information Networking Institute at Carnegie Mellon University, Pittsburgh, PA, USA• Adtranz, West Mifflin, PA, USA
Domain:	Maintenance
Purpose:	Support inspectors of a people mover system at the airport with a mobile computing system that offered communication means to the central office and other help desks
Challenges:	The maintenance and inspection had partially be done on the running people mover train, which runs in a tunnel beneath the surface
Device:	CMU’s “Adtranz” mobile pen-based computer
References:	[Siewiorek 1998], [Bennington 1999], [Bass 2001]

Description:

The Adtranz mobile computer received its name from the project partner of the CMU team in that project. Adtranz was at the time of the project a subsidiary of Daimler Benz. The motivation for this project was to reduce the downtime of the people mover system, which connects the landside of the airport with the airside, meaning the terminal and the gate areas. All passengers have to take the train in order to board a plane on an airside gate. In case of a downtime of one of the two trains, the capacity is cut by half, in case of a total breakdown of both systems all passengers have to either walk through an emergency tunnel or be transported by busses to the airfield. Flight delays and financial loss are results. Thus, this project aimed at developing a maintenance system with focus on the wireless communication between the

technician at the train and maintenance personnel at the central office or at other people mover locations that are installed around the world. The main challenges in the system design were the location of the maintenance activities (on a moving train in a tunnel) and the demands on the communication means (full-duplex voice and transmission of complex graphical data, such as blueprints).



Figure 7.10: (from left to right) Adtranz graphical user interface, Adtranz system in use, and people mover train in tunnel

Identified Work Locations:

Adtranz Locations		
1	Maintenance Shop Floor	On the shop floor, below the train
2	In tunnel	In tunnel around the train
3	Train (inside)	On the train (might be moving)

Table 7.22: Example locations of Adtranz project

Identified Work Activities:

Adtranz Activities		
1	Interact with Form	The technician has to retrieve, fill out, and submit the form for the right vehicle.
2	Use Electronic Manual	The technician retrieves information from an interactive electronic technical manual.
3	Inspect Object	The technician uses a tool and / or a flashlight for inspecting detail of the train.

Table 7.23: Example activities of Adtranz project

Implemented User Interfaces:

Adtranz Interfaces		
1	Rotary Dial	Input
2	Head-mounted Display	Output
3	Speech Recognition	Input

Table 7.24: Implemented user interfaces for Adtranz project

Discussion of used interfaces:

The special interface issues in this project were the need for online collaboration tools and thus communication means. This led to a design that included a video camera, and head-mounted display and full-duplex voice transmission via a wireless local area network. The system thus offered a fully featured videoconferencing and online help desk functionality that even worked in the tunnel on the moving people mover train.

7.10. OSCAR – Crane Operator Assistance

Project:	“OSCAR – Offshore Supply Crane Assisting Resource”
Research Team:	<ul style="list-style-type: none">• Institute of Complex Engineered Systems at Carnegie Mellon University, Pittsburgh, PA, USA• Chevron Corporation, Houma, LA, USA• Applied Hydraulic Systems, Houma, LA, USA
Domain:	Maintenance / Manufacturing
Purpose:	Assisting crane operators for lifting heavy weights from a supply boat to the oilrig where the sight of the operator is mostly blocked.
Challenges:	Harsh weather conditions, high accident risk, no direct sight to the load and the target for the load of the cranes.
Device:	Rugged Pentium-based PC and sensor beacons at the cargo and loading boat.
References:	[Oscar 1998], [Sackin 1999],

Description:

The computer system developed in the OSCAR project was aimed to support crane operators that have to move heavyweight cargo to and from a supply boat that is moving with the waves of the sea. The problem was that most of the time, the operator could neither see the load nor the boat on which the load was to be placed. Thus, the system aimed to transmit the position of the boat and the load to the crane operator at the cranes cabin or platform. Since these cranes sit on an oilrig, they are constantly affected by the weather conditions, such as rain, fog, or high winds, and other accident-prone factors, such as large heights and heavily moving boats, the highest goal for the system was to increase the safety of the participating workers. The system provided the location and orientation of boat and cargo load to the crane operator and thus provided feedback on the relative positions of load and target for the load. The system provided graphical feedback of this information on an LCD display that was connected to a rugged Pentium

PC located in the cabin of the crane. A video camera transmitted images of the load on the deck of the boat, which were incorporated in the graphical user interface.

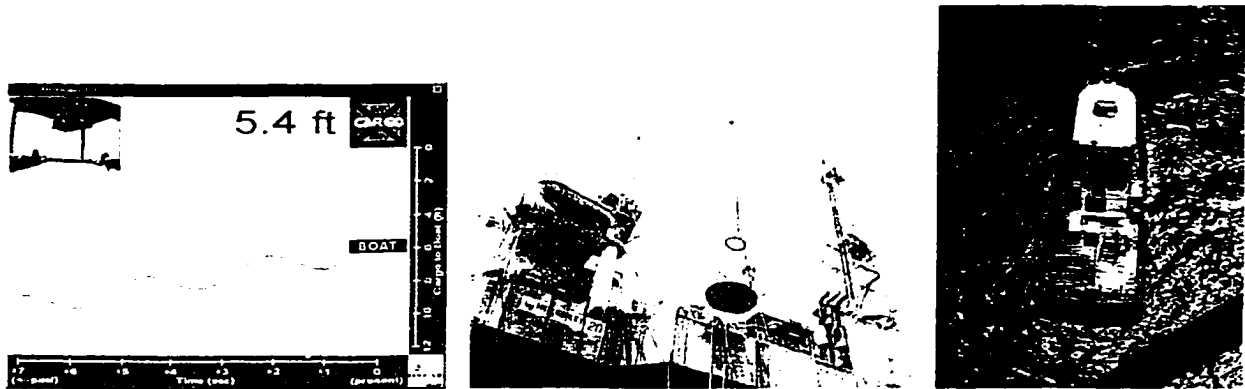


Figure 7.11: (from left to right) Oscar graphical user interface, crane with load, supply boat as target for the load.

Identified Work Locations:

OSCAR Locations		
1	Crane Cabin	In the cabin of the crane, use by the operator.
2	Crane Cabin (severe conditions)	In the cabin of the crane, use by the operator in severe weather conditions).
3	Boat Deck	On the deck of the supply boat, providing an assistance

Table 7.25: Example locations of OSCAR project

Identified Work Activities:

OSCAR Activities		
1	Lift Load from Boat	The crane operator lifts the load from the moving boat, no visibility.
2	Set Load on Target Point	The operator sets the load down on the oilrig.

Table 7.26: Example activities of OSCAR project

Implemented User Interfaces:

OSCAR Interfaces	
LCD Display	Output

Table 7.27: Implemented user interfaces for OSCAR project

Discussion of used interfaces:

In this project, the user interaction was kept at a minimum. All information that the crane operator would need could be shown in one display and updated constantly by wireless communication. Thus, there was no need for user input or additional output. This system could have incorporated additional feature that would have replaced traditional communication means, such as walky-talkies, or more complex systems, such as the communication means of the Adtranz project. But to not distract the crane operator could the project team decided to keep the system as simple as possible, which also had a positive effect on the maintenance of the hardware.

7.11. Stars 3 – Power Plant Maintenance

Project:	“Stars - Sticky Technology for Augmented Reality Systems”, 3 rd generation.
Research Team:	Joined software engineering course between: <ul style="list-style-type: none">• School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA• Department of Applied Software Engineering, Technical University Munich, Germany
Domain:	Maintenance
Purpose:	Support inspections of nuclear power plants in providing interactive electronic technical manuals (IETM) and Augmented Reality annotations and guidance within the facility to the technicians.
Challenges:	Establish a software architecture that allows for the delivery of the IETMs and the AR functionality and implement usable interfaces for both issues. The main challenges were to establish reliable networking capabilities and to display the 2D and 3D information in a usable way.
Device:	Wearable computer system with handheld and head-worn display.
References:	[Dutoit 2001], [Stars 2001], [Klinker 2001]

Description:

During the Stars project, the development team designed a software architecture for a wearable computer system that supports maintenance technicians of nuclear power plants. The mobile AR system provides information about components of the power plant. Components, such as pipes and valves that have to be inspected or maintained could be annotated either with inspection data, with the current flow in the pipe, or with arrows that show the location of the components. The interactive electronic technical manuals provided information about the technical details of the system, such as blueprints and specifications.

The software architecture of the system is divided into three subsystems: the IETM, the network and the user interface (UI) subsystem. The IETM subsystem handles all the data that the inspector needs in a specific situation. It retrieves the information from the IETM server and can then provide the information without further network connection. The network subsystem keeps track of the inspectors location and the status of the maintenance process and decides together with the IETM subsystem which data has to be prefetched to avoid waiting times for the file download. The UI subsystem controls all interactions with the user, such as the display output on the handheld display as well as the output for the AR system on the head-mounted display.

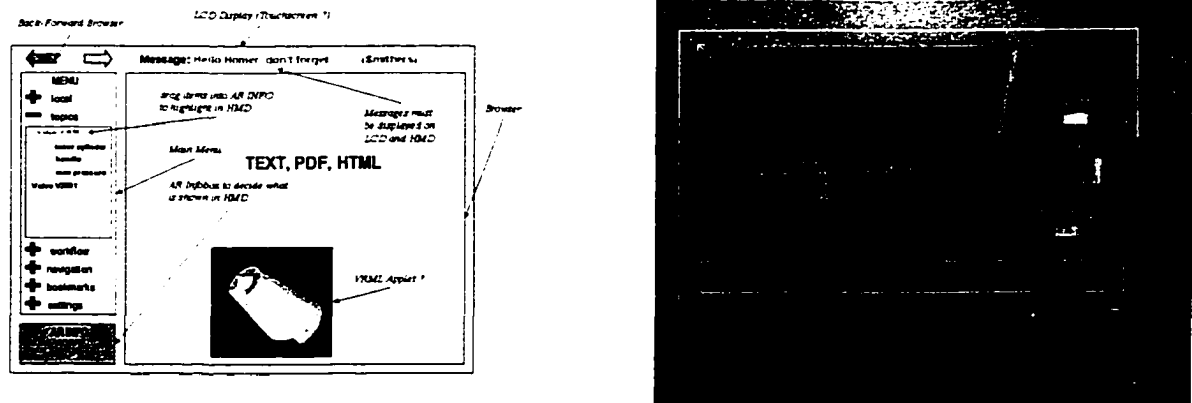


Figure 7.12: Stars 3 user interface prototypes for LCD display (left) and for the Augmented Reality system (right)

Identified Work Locations:

Stars 3 Locations		
1	Power Plant Floor	Within the building of the power plant.
2	Power Plant Facility (outside)	On the ground of the power plant at an outside inspection location.
3	Helium Flashing System	Near pipes and valves, which have to be inspected and maintained.

Table 7.28: Example locations of Stars 3 project

Identified Work Activities:

Stars 3 Activities		
1	Navigate with AR System	The inspector walks about the site and is guided by the AR system with arrows.
2	Use Electronic Manual	The technician retrieves information from an interactive electronic technical manual.
3	Inspect Object	The technician uses a tool and / or a flashlight for inspecting detail of the valves / pipes.

Table 7.29: Example activities of Stars 3 project

Implemented User Interfaces:

Stars 3 Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Head-mounted Display	Output
4	Speech Recognition	Input
5	Text-to-Speech	Output
6	Pointing Device at Wearable	Input

Table 7.30: Implemented user interfaces for Stars 3 project

Discussion of used interfaces:

This project illustrates the issues in integrating an Augmented Reality system in mobile IT support that delivers information based on the available network connectivity. It showed the challenges in the design of AR systems and interactive electronic technical manual and showed how to keep the user interaction reliable in client-server architectures even during disconnection from the network.

7.12. Winspect – Preventive Maintenance

Project:	“Winspect” a mobile system for maintenance and inspection of manufacturing machinery
Research Team:	<ul style="list-style-type: none">• TZI Bremen, Germany• Stahlwerke Bremen GmbH, Germany
Domain:	Maintenance
Purpose:	Support inspection and maintenance personnel of a steel manufacturing plant with a wearable computer system to cut down the maintenance costs.
Challenges:	Inspection personnel must inspect during the actual manufacturing processes and have to have their hands free for safety reasons
Device:	Xybernaut Mobile Assistant IV, with head-mounted display Proprietary data gloves
References:	[Boronowsky 2001], [Wearlab 2002], [Winspect 2002]

Description:

The Winspect project emerged from the enormous costs that the Stahlwerke Bremen GmbH, a German steel manufacturing plant has to spend on preventive maintenance of the hundreds of cranes that are installed in their facilities. These cranes are needed for the transportation of the steel products during and after manufacturing. A failure of a crane results in a delay of the manufacturing process. Without the support of the system the inspectors have to leave the crane that showed a faulty component and get to a safe location to take notes. Furthermore, if the fault requires using a specific manual of one of the different cranes, the inspector has to get this paper-based manual from a central location. The system was designed to keep the attention of the user to the environment free, to record findings about faulty crane components, to display large technical drawings, and to allow the inspector wearing protective gloves. The system offered a sensor-equipped glove, which could be used to select different

widgets on the graphical user interface and then scroll through lists of the crane components to select faulty parts. The same mechanism allows for browsing through technical documentation while wearing the protective gloves. Technical drawings can be navigated with the data glove by moving the active view port of the drawing with the motion of the gloves (see Figure 7.13).

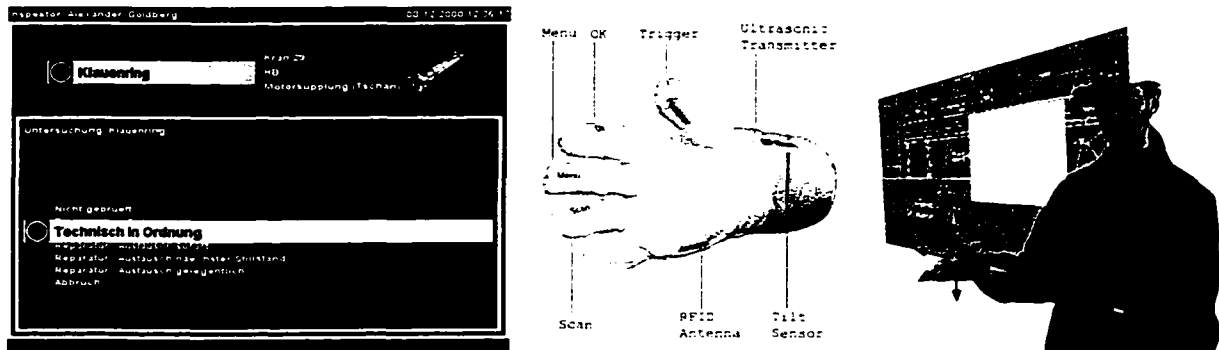


Figure 7.13: (from left to right) Winspect graphical user interface, Winspect sensor glove, and concept of handling large technical drawings.

Identified Work Locations:

Winspect Locations		
1	Manufacturing Shop Floor	On the shop floor, below the crane.
2	Crane	The technician climbs the crane to inspect some of the subcomponents.
3	Steel Mill	The technician moves near a steel mill within the plant.

Table 7.31: Example locations of Winspect project

Identified Work Activities:

Winspect Activities		
1	Interact with Form	The technician has to retrieve, fill out, and submit the form for the right crane.
2	Use Electronic Manual	The technician retrieves information from an interactive electronic technical manual.
3	Inspect Object	The technician uses a tool and / or a flashlight for inspecting detail of the crane.

Table 7.32: Example activities of Winspect project

Implemented User Interfaces:

Winspect Interfaces		
1	Sensor Glove	Input
2	Head-mounted Display	Output
3	Pointing Device at Wearable	Input
4	Head-mounted Display	Output
5	Video Camera	Input

Table 7.33: Implemented user interfaces for Winspect project

Discussion of used interfaces:

Winspect introduced an interesting concept for displaying large and complex technical drawings. With the data glove it offered a good alternative for pointing devices or speech navigation through lists and drawings, although it does not enable complete hands-free operation. However, the system showed how hardware user interfaces of mobile IT systems can be integrated into the protective clothing and thus ensures the safety of the user and the protection of the device itself.

7.13. Digital Hard Hat – Multimedia Field Data

Project:	“Digital Hard Hat” - Multimedia field data on construction sites
Research Team:	<ul style="list-style-type: none">• Department of Civil and Environmental Engineering, University of Illinois, Urbana-Champaign, IL, USA• U.S. Army Corps of Engineers, Fort Worth, TX, USA
Domain:	Construction
Purpose:	Enable collaboration on construction sites using video conferencing technology and multimedia data exchange methods.
Challenges:	Delivering multimedia technology to the construction site.
Device:	Fujitsu Stylistic with digital camera
References:	[Stumpf 1998], [CII 2002]

Description:

The Digital Hardhat system was developed to enable online collaboration between field engineers on construction sites and engineers at a remote office. The intention was to allow field engineers for getting feedback from colleagues on specific problems that cannot be decided by one single person on the construction site. Connecting field and office personnel establishes a possibility for ad hoc group decisions. This helps to reduce the number of instances, where a field engineer has to stop working to retrieve expert opinion or help on a specific problem or even to ask remotely located engineers to visit the site, which results in travel costs and possible downtime for the site. The Digital Hardhat offers a pen-based computer with an attached video camera and video conferencing software. Together with the developed software, called Multimedia Facility Reporting System (MFR), the Digital Hardhat allows for taking and annotating images of the site, transmitting video sequences to remote sites, and to talk to members of the project team. Recently the system was transferred to a Windows CE-based device, which offered

the benefits of smaller dimensions and a format that could be better carried around the construction site.



Figure 7.14: (from left to right) Digital Hardhat graphical user interface, annotated image of construction site, and system in use.

Identified Work Locations:

Digital Hardhat Locations		
1	Scaffolding	On scaffolding somewhere on the construction that is still in progress
2	Excavation Area	In an area, where excavation is going on and where the surface not yet leveled
3	Tunnel / Basement	In a tunnel construction or in a basement of a building construction.

Table 7.34: Example locations of Digital Hardhat project

Identified Work Activities:

Digital Hardhat Activities		
1	Interact with Other People	The field engineer annotates images and exchanges
2	Interact with Photo	Sometimes picture proofs are required.
3	Move about Inspection Site	The engineer has to walk / climb the site and check the structure at different locations.

Table 7.35: Example activities of Digital Hardhat project

Implemented User Interfaces:

Digital Hardhat Interfaces		
1	Handheld Display	Output
2	Touchscreen	Input
3	Digital / Video Camera	Input

Table 7.36: Implemented user interfaces for Digital Hardhat project

Discussion of used interfaces:

The Digital Hardhat system showed the interaction with engineers at remote locations and demonstrated the applicability of multimedia technology to construction sites. Thus, the user interfaces of the system must allow for easy annotation of the transmitted images to ensure the timely transmission during the online collaboration. Furthermore, during online audio conferences, speech-controlled operation of the system would distract the other participants. Thus, in this case the use of the pen-based computer with its touchscreen was a sufficient solution.

7.14. NOAH – Aid for Emergency Physicians

Project:	“NOAH” - Emergency Organization and Administration Aid.
Research Team:	<ul style="list-style-type: none">• Department of Trauma Surgery University of Regensburg Medical Centre, Germany• Chair of Business Informatics III at the University of Regensburg, Germany• Kratzer Automation AG, Unterschleißheim, Germany
Domain:	Emergency Response
Purpose:	Support emergency response teams in documenting and transmitting the initial examination of patients.
Challenges:	Documentation of on-scene data has to be as complete as possible and be transmitted to the hospital in a timely manner. The documentation process itself must be time-efficient to allow more therapy time.
Device:	Xybernaut Mobile Assistant IV Motorola Forte, pen-based computer
References:	[Schaechinger 2000], [Noah 2002], [Roeckelein 2002]

Description:

NOAH is a system that supports on-scene physicians of emergency response teams in the initial documentation of patient data. It allows for a completely electronic documentation of information about the kind of injury that occurred and the therapy that has to follow in response to the injury. Thus, the data can be processed and transmitted electronically and the corresponding preparations in the hospital can be initialized, respectively the hospital can be chosen based on the availability of beds and needed equipment. Two different hardware systems were used for this project: A Xybernaut MA IV wearable computer and a pen-based computer from Motorola. The system allowed for a faster transmission of the injury data to the hospital than with the use of audio radio transmission. This saves time for further therapy and preparation in the hospital. The

software allows to immediately entering information about the patient and the injury on a graphical user interface and, in case of a not easily recognizable injury or symptom, such as poisoning, it provides an analysis tool for identification of the right treatment on the scene. Due to the nature of the application, the scene of the accident can be any imaginable situation and thus with several kinds of constraints.

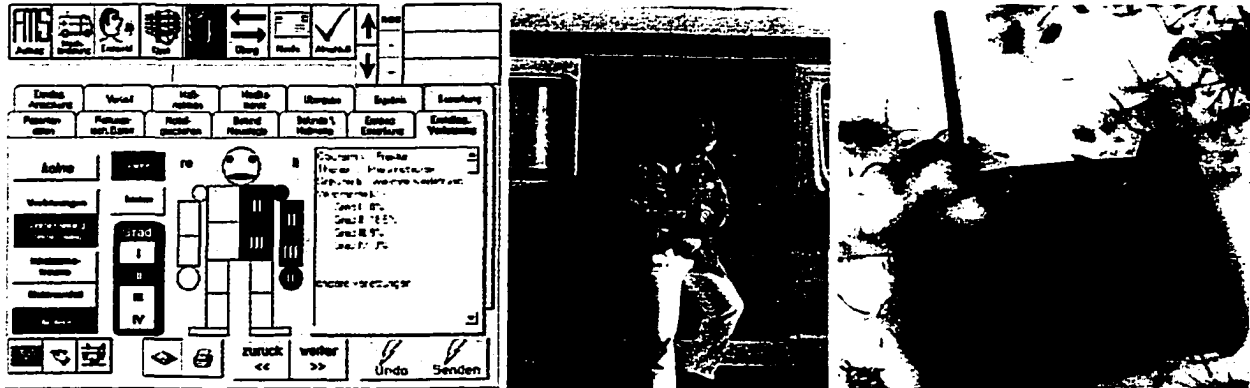


Figure 7.15: (from left to right) NOAH graphical user interface; NOAH in use on Xybernaut MA IV and on Motorola Forte

Identified Work Locations:

NOAH Locations		
1	Apartment	The physician arrives at the patient's apartment.
2	Highway (heavy traffic)	At an accident scene during commuter traffic.
3	Highway (at night)	At an accident scene at night.
4	Vehicle Wreckage	At an accident scene in which the patient is stuck in the vehicle.
5	Sports Arena	An athlete was injured and the physician enters the field.

Table 7.37: Example locations of NOAH project

Identified Work Activities:

NOAH Activities		
1	Interact with Form	The physician fills in the patient data.
2	Use Medical Database	The physician retrieves information from the database on the device.
3	Interact with People	The physician has to talk to colleagues and / or the patient.
4	Reanimate Patient	The physician reanimates a patient.

Table 7.38: Example activities of NOAH project**Implemented User Interfaces:**

NOAH Interfaces		
1	Touchscreen	Input
2	Handheld Display	Output
3	Pointing Device at Housing	Input

Table 7.39: Implemented user interfaces for NOAH project**Discussion of used interfaces:**

The physicians preferred the pen-based computer because it better reflects the analogy to the formerly used clipboard and pen [Roেকেlein 2002]. Problems in the interaction with the device occur 1) if the physician has to talk to patient, if they need psychological support, 2) if the physician has to reanimate or otherwise treat the patient with both hands, or 3) if discretion of some data has to be kept. Furthermore, the user interaction has to reflect the time-sensitive process and the need for reliability, caused by the seriousness of a failed therapy.

7.15. Deep Map – A Virtual Tourist Guide

Project:	“Deep Map” A Virtual Tourist Guide in Heidelberg
Research Team:	<ul style="list-style-type: none">• European Media Laboratory GmbH, Heidelberg, Germany• Fraunhofer IGD, Darmstadt, Germany,
Domain:	Maintenance
Purpose:	Guide tourist through the historic town of Heidelberg, Germany while providing context-sensitive information about the tourist attractions.
Challenges:	Provide the data for the context-sensitive information and browse through these multi-lingual databases based on natural language processing to support tourist of different nationalities.
Device:	Xybernaut MA IV IBM laptop
References:	[Coors 1999], [Malaka 1999], [Elting 2002]

Description:

The Deep Map system provided several functionalities to the user. The first one is a “talking map” that guides the user through the city and answers questions that can be entered in natural language with a limited vocabulary. Thus system adapts to the location of the user and recalculates the route for the tour. The system is based on general Global Positioning System (GPS) technology that allows for location-aware adaptation and context-aware information for the user. The second feature is a virtual map of Heidelberg, Germany, which offers 2D as well as 3D illustration of the city. The user can take virtual tours of the city and even virtually fly through the city in the 3D map. Finally, the system offers a multi-lingual speech recognition engine and natural speech processing capabilities, which allows for entering database inquiries in English, German, and Japanese. Besides the speech interaction the system offers a handheld display of the wearable computer system, which enables the user to see a map or video sequences of the

city. For a more complete information system, the deep map can be connected to a hotel reservation system to allow for online reservation and guidance between tourist attractions and the hotel.



Figure 7.16: (from left to right) Deep Map graphical user interface, Deep Map system in use, and envisioned use on a PocketPC device

Identified Work Locations:

Deep Map Locations		
1	Hotel	The tourist is at hotel and starts the city tour.
2	Pedestrian Area	The tourist walk along the pedestrian area in Heidelberg.
3	Bar	The tourist rests in a restaurant or bar.
	Museum	The tourist takes a tour in a museum.

Table 7.40: Example locations of Adtranz project

Identified Work Activities:

Deep Map Activities		
1	Retrieve Guidance	The tourist retrieves guidance on the tour through Heidelberg.
2	Use Electronic Manual	The tourist retrieves information about tourist attractions.
3	Interact with People	The tourist contacts friends or a hotel.

Table 7.41: Example activities of Adtranz project**Implemented User Interfaces:**

Deep Map Interfaces		
1	Speech Recognition	Input
2	Speech Synthesis	Output
3	Handheld Display	Output
4	Touchscreen	Input
5	Pointing Device at Wearable	Input

Table 7.42: Implemented user interfaces for Adtranz project**Discussion of used interfaces:**

In the Deep Map system the databases and the contained data were the key features. The important user interface for the system were the speech recognition engine and the natural language processing, which allowed for multi-lingual natural speech input to retrieve information from the databases. Although the system still needed two separate computers to establish the needed processing power, which might be solved by emerging hardware components, it showed another example of user interaction, which might be a key technology in the future.

8. Illustrative Usage Example

Based on the implementation of the *Interaction Constraints Evaluation Tool (ICE-Tool)* described in Chapter 6 and the data collected from real projects described in Chapter 7, this chapter will present a complete usage example of *ICE-Tool* being used on a new project. The example will illustrate how to identify *work situations*, i.e., *work activities* and *work locations*, and how to set up the corresponding sets of *constraints* for *work situations*. The example will also describe how *ICE-Tool* queries for similar sets of *constraints*, and how user feedback of specific projects can be entered into the database.

For the sake of an illustration, I chose an application with *work situations*, which are dissimilar to those of the projects described in the previous chapter that already exist in the *ICE-Tool* database. This ensures that the example is independent from any other projects that were previously entered in the system. The application chosen, named the “Automated Daily Log”, supports the management and evaluation of daily construction reports, which are composed by the general construction superintendent and all subcontractors and then merged by construction management staff. It focuses on what and who is present at the site at a specific point in time, rather than capturing the progress of the construction process (which is the objective of the ICMMS project described in Chapter 7.4). The purpose of this *Automated Daily Log* application is to collect a daily snapshot of what is going on at the site to help the construction management in verifying the reports of the subcontractors. This additional view of the site helps to resolve cases in which reports of the subcontractors show ambiguous data or claims.

The *Automated Daily Log* application was designed as an *m/w-CAE System* by a student project team of a product design course at Carnegie Mellon University. During the design process, I worked with the project team to collect the necessary *constraints* to provide feedback about possible user interfaces that the team envisioned. The *Automated Daily Log* was designed for and field-tested at the new David L. Lawrence Convention Center in Pittsburgh, PA, USA.

The *Automated Daily Log* has several subsystems supporting the construction management as well as the different subcontractors. This example covers the user interaction

design for the *m/w-CAE System* that will support the construction management staff: currently, a member of the construction management staff periodically performs the activity by carrying a clipboard and a digital camera to capture situations that need to be documented. Therefore, a digital camera will be connected to the envisioned *m/w-CAE System*. The *m/w-CAE System* stores pictures based on the needs of the application either directly on a hard drive or in a database and allows for annotations by the user.

The usage scenario can be described as follows: a member of the construction management staff walks about the construction site and collects information about 1) which machinery is currently at the site and which of it is in use, 2) which building components or construction materials were delivered and are located at which location and in which condition, and 3) how many workers (roughly) are present and what are their current activities.

The scenario described here contains two activities supported by the *Automated Daily Log*, where 1) the construction manager takes a picture of a misplaced stack of delivered window frames and 2) the construction manager takes notes about a sample taken from a concrete delivery. The generic use case for these examples are “*document information with picture*” and “*document information about delivery*”. Figure 8.1 shows the simplified use case diagram. Note that there are many more use cases that I do not cover in order to make this example more comprehensible.

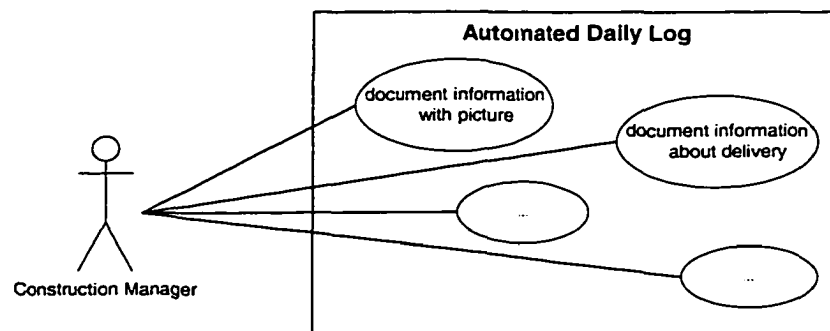


Figure 8.1: Use Case Diagram showing the two example use cases

8.1. Identifying Work Situations

To identify and analyze *work situations* of the example use cases, we have to perform three steps: 1) perform a task analysis and draw the corresponding *UML Activity Diagrams*; 2) identify locations on the site at which the system will be used; and 3) enter the *work activity* data and *work location* data into the system.

8.1.1. Identification of Work Activities

The first step to use the system is to perform a task analysis as discussed in Chapter 4, which helps to identify the *work activities* and the need for user interfaces and interfaces to other components of the IT infrastructure. Using swimlanes in *UML Activity Diagrams* to identify these interfaces at each change of swimlanes has proven useful in previous projects [Buegry 2000, p. 27-39], [Meissner 2001, p. 62-65]. Each change of swimlanes indicates that the task flow in the *Activity Diagram* transfers from one component or *Actor* to another, which results in the need for some kind of interface. This visualization seems to be the best way for determining work situations that need to be investigated for the design of user interaction with *m/w-CAE Systems*.

Figures 8.2 and 8.3 show the *Activity Diagrams* for the *Automated Daily Log* activities. Figure 8.2 shows swimlanes for the user, the *m/w-CAE System*, and the digital camera; Figure 8.3 shows swimlanes for the user, the *m/w-CAE System*, and a central database system. The numbers in the circles refer to a swimlane change. Thus, each number represents either a user interfaces or an interface to another system component. As such, these numbers serve as identifier for these interfaces during the design process. In the case of the interaction with the camera, it is easy to reproduce the need for a graphical user interface at interface #004, because the system presents the preview of the digital images to the user. Similarly, interfaces #109 and #110 in Figure 8.3 represent interfaces to the central database and thus indicates the need for network connection between the *m/w CAE System* and the database server. Both diagrams only show parts of the task analysis necessary for the interaction design for the *m/w-CAE System* that will run the *Automated Daily Log*.

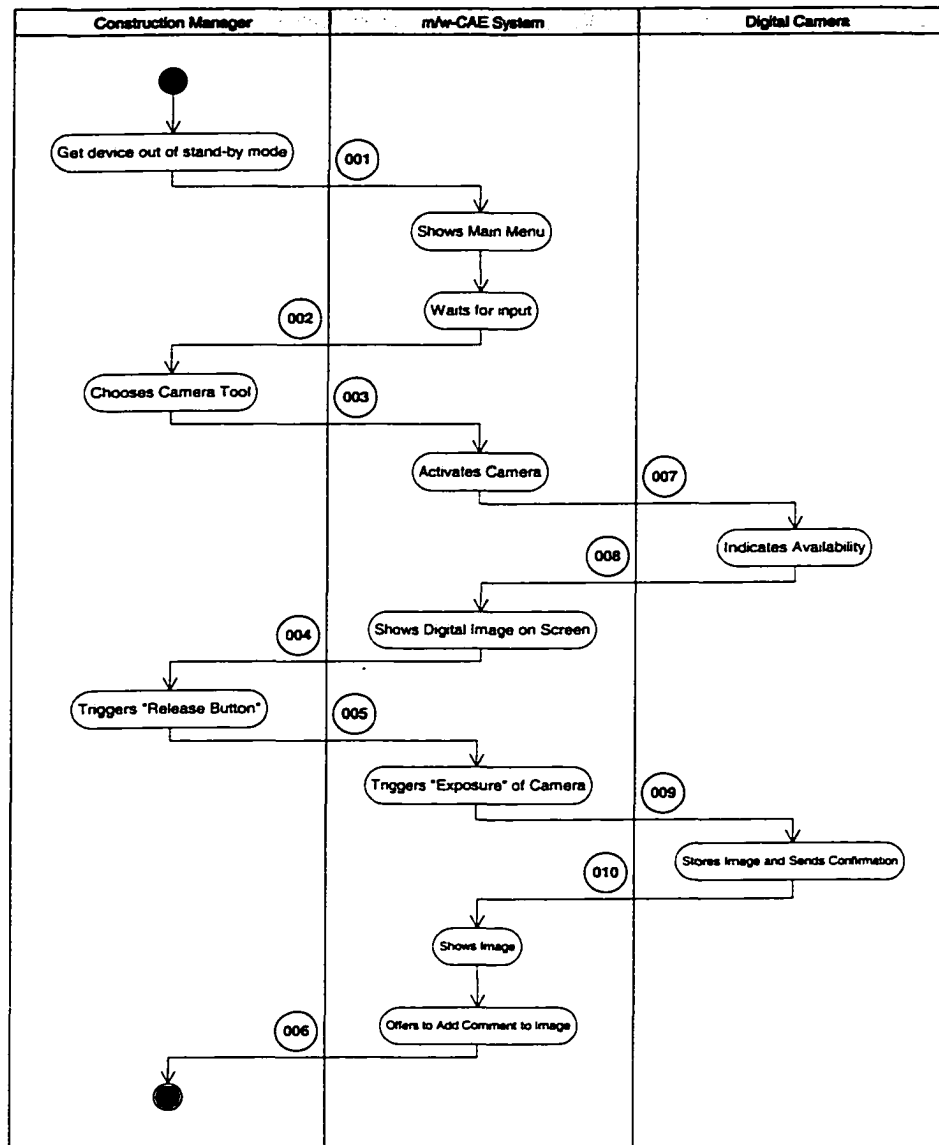


Figure 8.2: Activity Diagram of Task “document with picture”; User Interfaces are indicated by swimlane changes #001-006, interfaces to the digital camera are required at #007-010

The *Activity Diagram* in Figure 8.2 shows that six user interfaces have to be designed for the swimlane changes #001-006, where odd numbers indicate user input and even numbers show the need for feedback from the *m/w-CAE System*. Changes #007-010 indicate interfaces to the digital camera. The camera indicates its status to the device and allows for reviewing and storing the images on the *m/w-CAE System*.

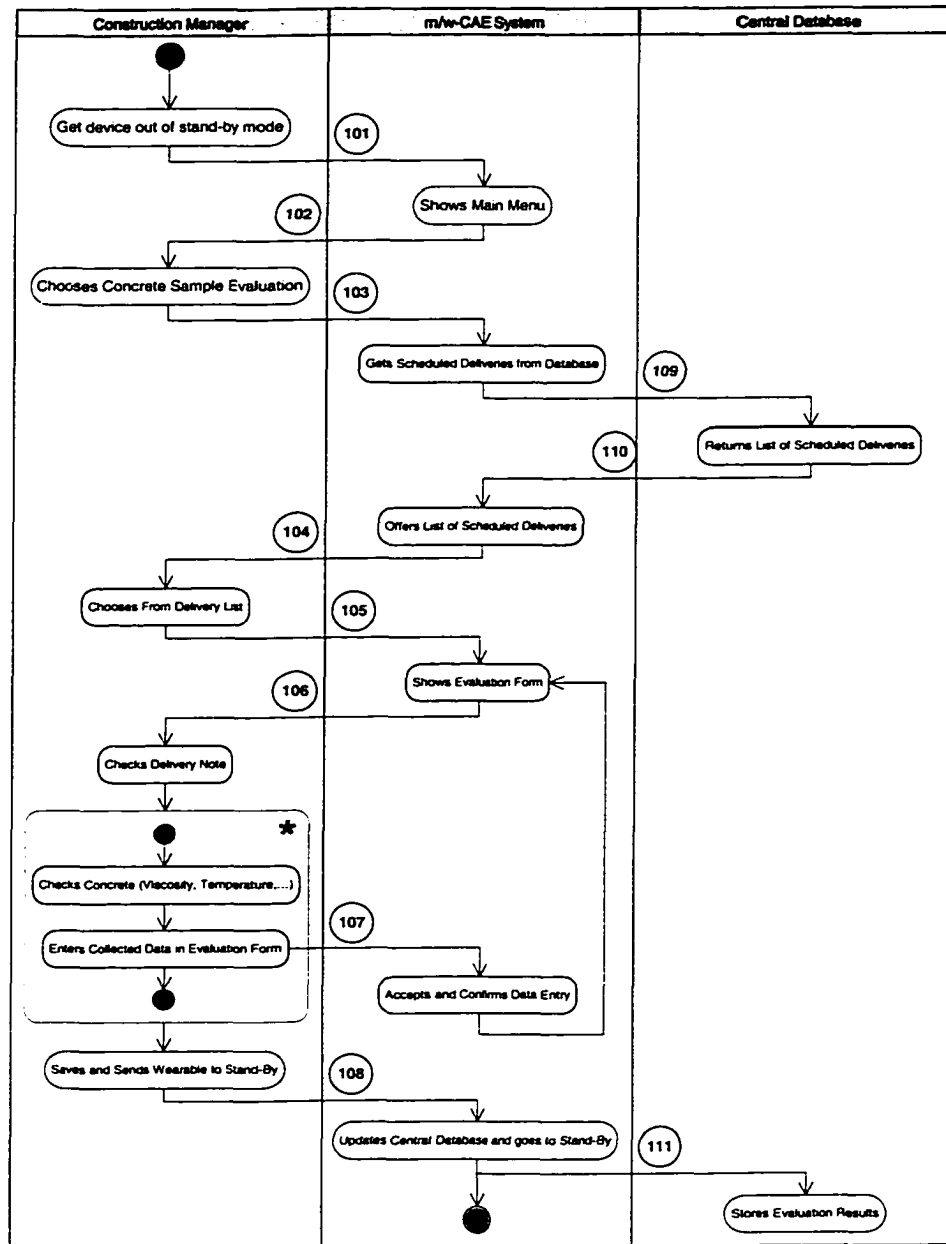


Figure 8.3: Activity Diagram of Task “document information about delivery”; User Interfaces are indicated at swimlane changes #101-108, interfaces to the digital camera are required at #109-111.

Figure 8.3 shows another *UML Activity Diagram*, which illustrates that “Checks Concrete” is an iterative task (or a “Dynamic Concurrency” in UML terminology) indicated by the multiplicity marker (*) [Fowler 1999, p. 135] and consists mainly of *Primary Tasks*, i.e., no support by the computer is necessary until the results are entered into the system. Although the actual process

of checking concrete has several more *Primary Tasks*, it is acceptable to skip these, since *Primary Tasks* do not need interaction with the device (see Chapter 5.2). However, if special testing equipment or new techniques for the concrete testing would be necessary, for example for a new High-Performance Concrete, the construction manager might want to see an interactive description from an electronic user manual that provides guidance through the testing process. This would be an additional use case and would be covered by additional *Activity Diagrams*.

To identify the actual *work activities* that are related to interaction between the user and the *m/w-CAE System*, we analyze the swimlane changes in both diagrams and decide on the kind of interaction that is done at the specific task. In the examples shown in Figures 8.2 and 8.3, we can identify *interaction with lists* for interfaces #002, #003, #006, and #102-105; *interaction with images or photographs* for interfaces #004 and #005; and *interaction with forms* for interfaces #106-108. Interfaces #001 and #101 can also be considered to be *interaction with lists*, although there might be only one option in this list or menu – putting the device into stand-by mode.

Since it does not make sense to divide the application into too many *work activities*, which would lead to multiple entry of the same *constraints*, we will only use the categories we just identified: *interaction with lists*, *interaction with photographs*, and *interaction with forms*.

8.1.2. Identification of Work Locations

The identification of *work locations* should be either done on the actual site, based on previous site visits, or by interviewing people, which are working on this site. If this assessment has been completed before for another application and the *work location* is already in the system, this step can be skipped completely. For this example, I will describe four different *work locations* from the construction site of the new David L. Lawrence Convention Center. Certainly, there are more locations that could be identified on such a big construction site, but showing four of them is sufficient to follow the course of this example. Chapter 7 illustrated that showing 3-5 distinctive

work locations and 3-5 distinctive *work activities* describes the *constraints* for an application very well. The figures shown below are taken during visits to the construction site.

The first two *work locations* relate to the construction manager moving about the construction site. Construction sites are prone to have objects and construction material lying around and blocking the walkway for the workers. People on construction sites have to watch their step and thus should not be limited in their perception. Note that even if the device is not in use, it can still influence the user's perception. For example a head-worn display could block one of the user's eyes. The pictures in Figure 8.4 show a user of a mobile computer on two different kinds of staircases. The first one is located within the building and offers sparse artificial lighting; the second one leads to an outside portion of the construction and thus is in direct sunlight. Although both locations are similar for the worker's *Primary Task* of walking up or down the staircase, the *constraints* on the interaction with the device differ greatly.

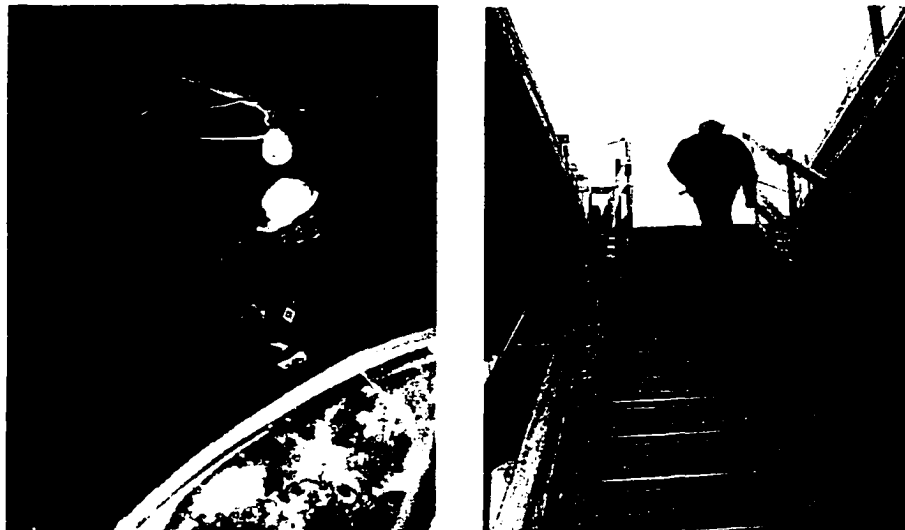


Figure 8.4: Two different kinds of staircases: inside the construction (left) and leading to the outside of the construction (right).

The third *work location* links to the construction manager's task of checking machinery. One such *work location* is in a cherry picker, which offers limited space and needs the driver to

use both hands to steer and elevate the basket. Furthermore, most of the tasks that are performed in the cherry picker are over-head, occupying both hands. Therefore, this is another meaningful example of a *work location*. Figure 8.5 illustrates the narrowness of the workbasket of the cherry picker, especially for the use of blueprints and other paper-based documents. Figure 8.5 also illustrates and how workers are elevated and restricted in their movement during their work. This example *work location* illustrates very well which advantages mobile IT support can offer. If we can retrieve any piece of construction documentation at the *actual work location*, then we can avoid moving back and forth between this location and a central office or kiosk-like central computer, which is especially interesting if the location is the basket that takes several minutes to be elevated and moved to the exact position.

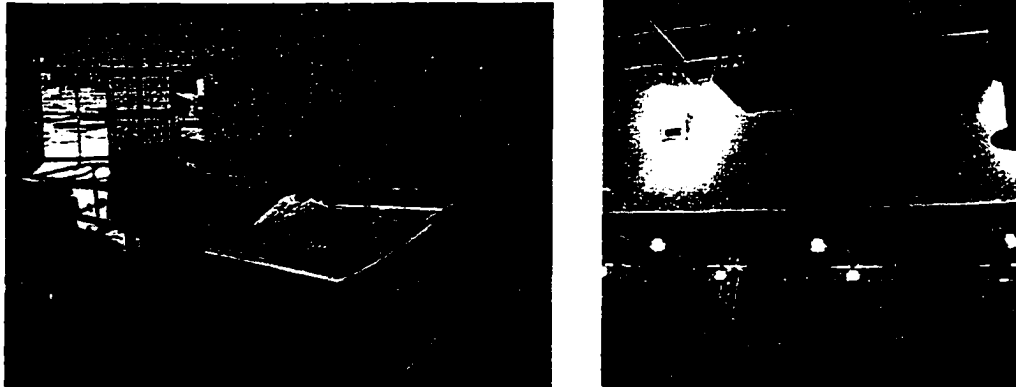


Figure 8.5: Cherry picker workbasket with blueprints (left) and cherry picker with elevated basket (right).

The last *work location* for this example relates to the concrete testing done on the construction site. Figure 8.6 shows the location at which a concrete sample might be taken. Although the actual conditions might differ in some details, inspectors will always find some machinery in use, which cause noise. They will also have to deal with dust, water, and concrete

Chapter 8: Illustrative Usage Example

being splashed on them when at this location. Thus, if the device is to be used in this work location, the ruggedness of the device has to be increased in terms of “Ingress Protection,” a standard for protection of electronic equipment against solid objects and liquids [TUV 2002]. On the other hand, it might be more efficient to design the user interface in a way that the device can stay in a protective pouch or in the user’s pocket while carried around this location. This however, depends on the *work activity* that is intended to be performed at this *work location* and thus depending on the whole *work situation*.

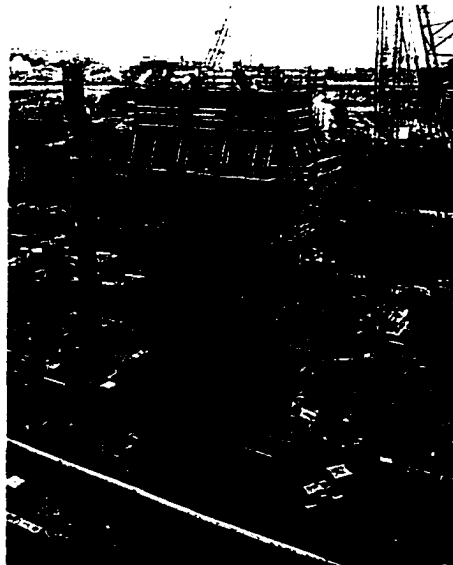


Figure 8.6: *Work location* involving dust, water and concrete splashes.

8.1.3. Data Entry

After identifying the work activities and work locations, the next step is to enter this data in to the *ICE-Tool* system. Therefore, the designer can select the “new location” and “new activity” buttons next to the corresponding list boxes in the main window. Dialogs will open, which allow for specifying names for the activities and locations and lets the designer enter a short description of each component.

Figure 8.7 shows the main screen of the *ICE-Tool* implementation as described in Chapter 6. The combination of selected *work location* and selected *work activity* identifies the *work situation* to which the information about *constraints*, their *influences* and implemented *user interfaces* are linked.

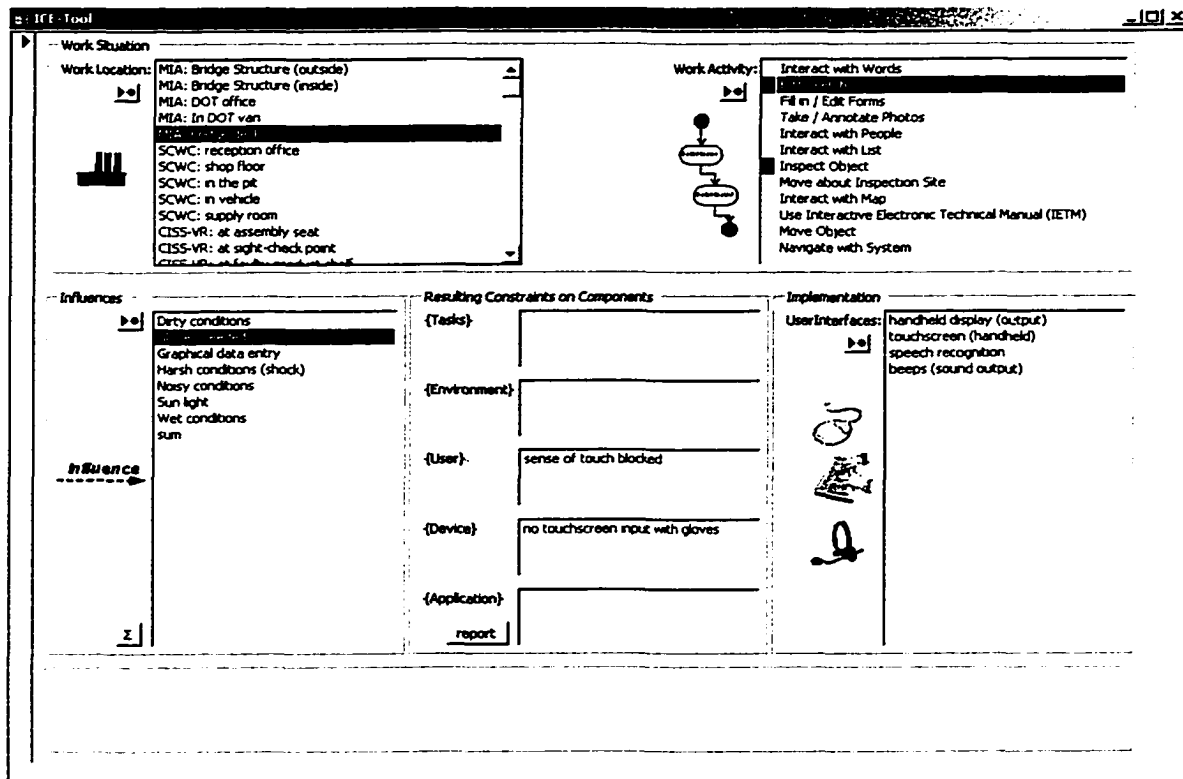


Figure 8.7: Main Screen of the *ICE-Tool* as implemented with MS Access. *Work location* and *work activity* list boxes are located on the top of the screen, the *influence*, *constraint* and *user interface* lists at the bottom.

Figures 8.8 and 8.9 show the entry dialogs for the *work location* and *work activity* taken from the actual entry of the example data. With these dialogs, the four work locations and three work activities identified for this example can be entered.

The dialog box 'New Location' has a title bar with standard window controls. It contains two text input fields. The first field is labeled 'Name' and contains the text 'ADL Staircase inside'. The second field is labeled 'Description' and contains the text 'an inside staircase from the Construction Site Inventory project'. Both fields have small upward and downward arrow icons on their right sides, indicating they are scrollable.

Figure 8.8: Location input dialog

The dialog box 'New Activity' has a title bar with standard window controls. It contains two text input fields. The first field is labeled 'Activity' and contains the text 'Interact with photos'. The second field is labeled 'Description' and contains the text 'The user has to retrieve or take and send photographic information'. Both fields have small upward and downward arrow icons on their right sides, indicating they are scrollable.

Figure 8.9: Activity input dialog

At this point, we entered the four *work locations* (two kinds of staircase, cherry picker, and a concrete mixer) and the three kinds of *work activities* (interacting with lists, photos, and forms), and can now proceed to enter the *influence* and *constraint* data and in doing so actually define the *work situations*.

8.2. Definition of *Influences* and *Constraints*

Influences can be seen as the source of *constraints* and as such are strongly related to them. In the *Interaction Constraint Model*, *influences* emerge from specific *work situations*. Some *influences* might only result from either the *work location* or from the *work activity*, but many result from the specific combination of both components. Thus, to enter *influences*, the system provides a list of previously entered *influences* to be reused (see Figures 8.10 and 8.11). Although these *influences* themselves can be similar for different *work situations*, the resulting *constraint patterns*, which are divided into the five *constraint categories* (*task, environment, user, device, and application*), are specific for each *work situation*. Thus, we have to first enter the *influence* that occurs for a specific *work situation* (defined by the selected items in the location and activity list boxes) and then enter the *constraints* that result from this *influence* into the appropriate *constraint category*.

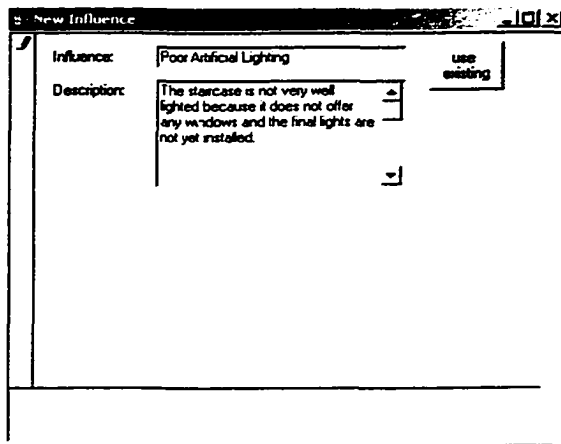


Figure 8.10: Influence input dialog

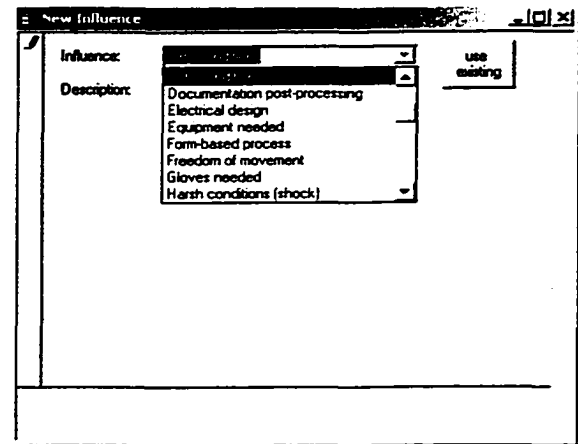


Figure 8.11: Influence input dialog with drop-down list of previously entered influences

Figures 8.12 to 8.16 show the entry dialogs for *constraints* for each of the five *constraints categories* (*task environment, user, device, and application*). These *constraints* are related to different *influences*. The data fields on each dialog related to the *Interaction Constraints Model* as discussed in Chapter 5 (see Tables 5.1- 5.5).

Figure 8.12 shows the input dialog for *task constraints*. The reduced visibility is caused by the *influence* “poor artificial lighting” in the inside staircase. Walking and operating the device are *Primary* and *Secondary Tasks*, and thus, the corresponding boxes are checked. Since the construction manager might use a flashlight in this poor artificial lighting condition, the attribute “Tools needed” is checked.

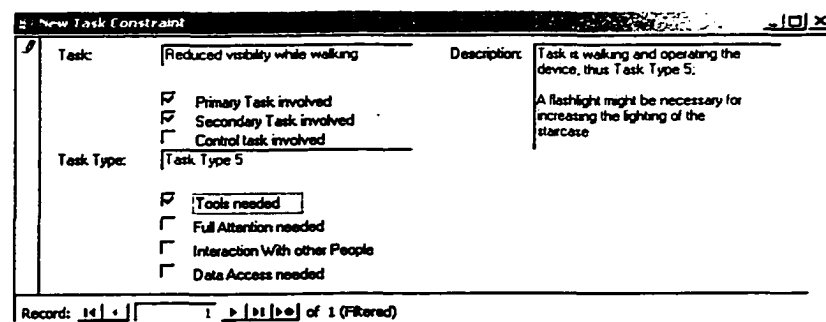


Figure 8.12: *Task constraint*, related to *influence* “Poor Artificial Lighting” (caused by the closed staircase).

recognition rate of lip or gesture recognition engines. No other attributes are affected by this influence.

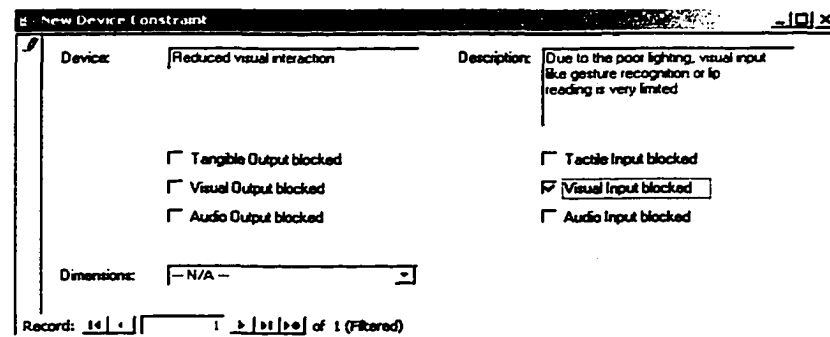


Figure 8.15: Device constraint, related to influence “Poor Artificial Lighting” (caused by the closed staircase).

Figure 8.16 shows the input dialog for *application constraints*. The influence “Form-based process”, which reflects the process of entering data into an incident report form, results in the two attributes “Text-based data” and “Table-based data” being checked. The attributes that describe the multimedia data are not affected by the influence.

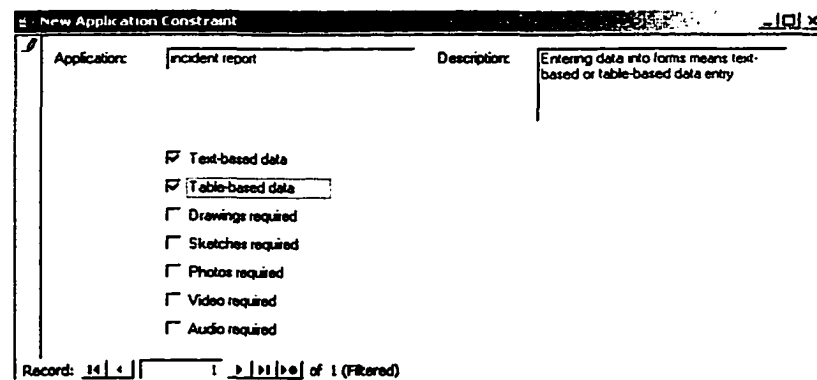


Figure 8.16: Application constraint, related to influence “Form-based process” (caused by the nature of the incident report).

The idea of separating the different *constraints* into categories becomes useful when one *constraint* can be compensated by another *constraint*, such as a noise-canceling microphone attached to the *device* could compensate for the user’s blocked linguistic ability. Also, if the protective ear-plugs provide built-in speakers, the aural cognition of the user (for the interaction

with the device) is better than without this feature. Thus, to get the actual limitations of all *constraints*, we have to combine all different *constraints* caused by all occurring *influences* (see the next section for further discussion).

8.3. Evaluation of *Constraints*

The goal of evaluating the *constraints* of a *work situation* is to find other *work situations* in the database that are described by a similar *constraint patterns*. To evaluate the *constraints* and map the set of *constraints* of a specific *work situation* to similar patterns of previously entered *work situations*, the combination of *constraints* must be considered (i.e., to combine the effects of different influences) into a set of five *constraints* - one *constraint* per category - that represent the most restrictive set of *constraints*. If for example the ambient noise restricts the user's aural cognition and the ambient light restricts the user's visual cognition, the combination or "sum" of both *constraints* will be one *user constraint* that restricts both the aural and the visual cognition.

In the ICE-Tool implementation clicking the *sum* (Σ) button will "sum up" the *constraints*, i.e., it will compile a union or combination of the *constraints* in each category. This will create a new *influence* called "sum", which has one entry in each *constraint category*. These entries contain the most restricting set of *constraints* generated from all *constraint* entries in the list. In the current version of *ICE-Tool*, this functionality only switches each attribute to the most restrictive value of all *constraints* of a *work situation*. Based on these five most restrictive *constraints*, the system can generate a report that shows *work situations* (i.e. combinations of *work locations* and *work activities*) of previous projects that were implemented under the same set of five *constraints*. Figure 8.17 shows the *sum* functionality for *work situations* at the *work location* "Staircase inside" and the *work activity* "Interact with list". The "sum" entries in the five *constraint categories* are the most restrictive *constraints* for the *influences* "Poor Artificial Lighting", "Uneven Steps" and "Form-based process".

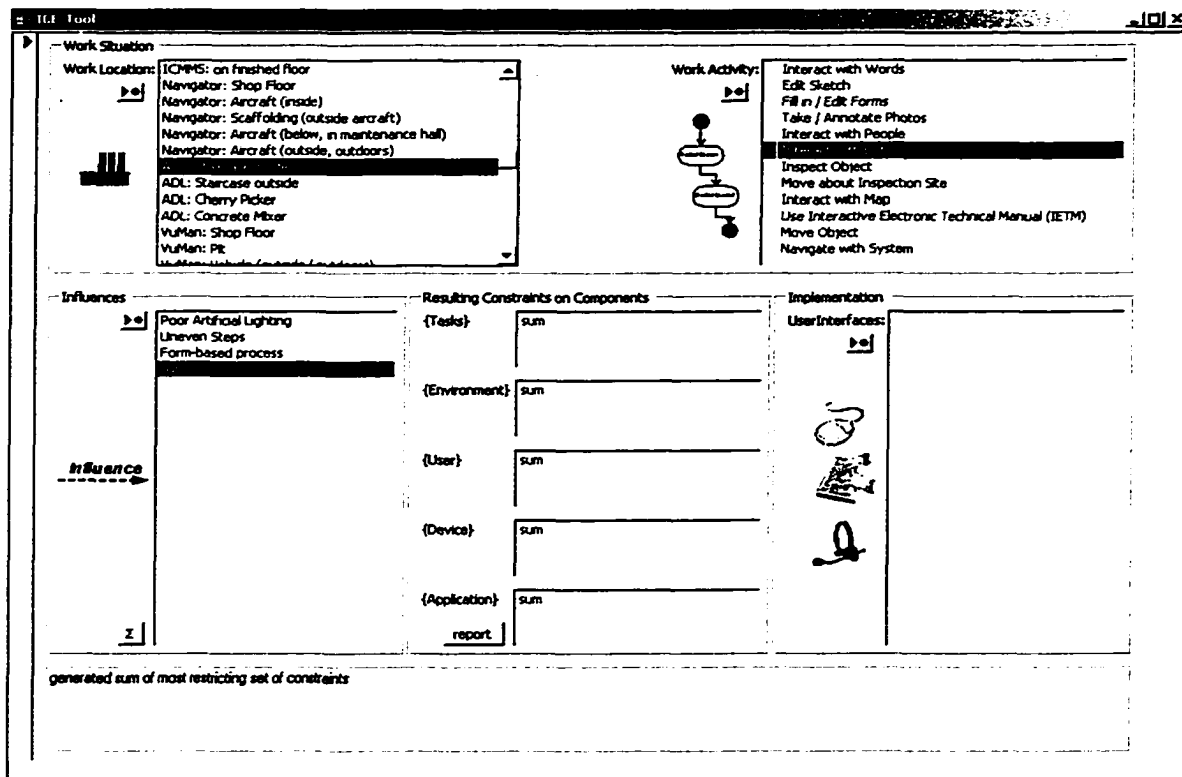


Figure 8.17: Application constraint, related to influence “Form-based process” (caused by the nature of the concrete testing).

ICE-Tool allows a shortcut for the data input process by allowing the omission of the definition of separate *influences*. Users of *ICE-Tool* can start with one *influence* named “sum.” In the five *constraint categories*, the user can then enter single *constraints* that describe the specific *work situation*. Thus, the data entry process can be shortened with the cost of reduced information about the causes of the *constraints*.

Based on these most restrictive *constraints* of the selected *work situation*, *ICE-Tool* can generate different reports that list the *work situations* with the same set of *constraints* in a specific *constraint category*, or the *work situations* that have the same set of *constraints* in all categories. Hence, to find a matching situation, the *constraint attributes* have to be set to the same values. For this example, Figures 8.12 - 8.16 present these values for the different categories. Figure 8.18 lists all *constraints* and the selected values for the list-processing task in the staircase that was covered before during this example.

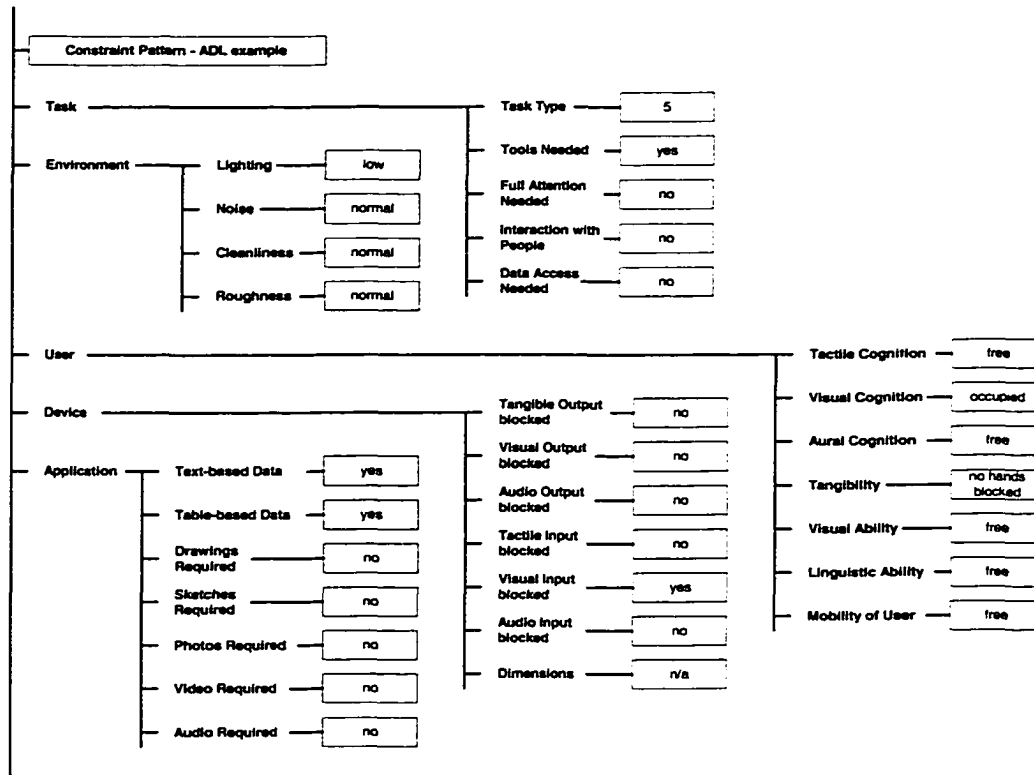


Figure 8.18: Constraints for staircase work situation from ADL example.

Figure 8.19 shows a dialog in which *ICE-Tool* offers to select which report to generate and calculates the numbers of matching *work situations* for each *constraint category*. In the case of the inside staircase of the *Automated Daily Log* example, the system found two *work situations* with the same set of *constraints*, i.e., one additional situation with the same set of *constraints*.

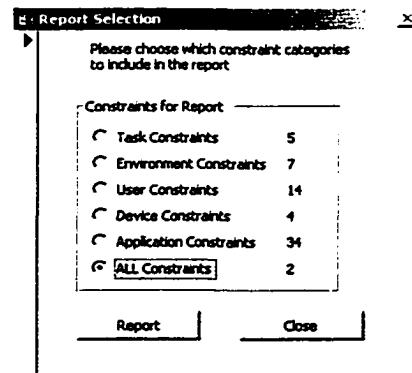


Figure 8.19: Reports selection dialog. The numbers indicate work situation with the same set of constraints in the specific constraint category.

A report based on all *constraints* for this *work situation* is shown in Figure 8.19. This report shows the *work location* and the *work activity* for matching *work situation*: the *Automated Data Log* situation itself, and a situation from the SCWC project in which a form is filled in at the supply room. With this information, the user of *ICE-Tool* can go back to the main screen, select the specific situation and retrieve the information about implemented user interfaces (the steps for retrieving the user interface is shown in Section 8.4). Future versions of *ICE-Tool* might offer a more detailed report, which itself can contain information about the user interface implementation for the matching *work situations*. Appendix A lists the *work situations* that are generated by the different single *constraint categories*, which gives an idea of which *work situations* are similar with respect to the selected *constraint category*. The high numbers of matching situations result from fewer *constraints* in the specific category. For example the device category was not at all constrained in this situation, since no specific dimensions for the envisioned device were given.

Work Location	Work Activity
ADL Staircase inside	Interact with List
SCWC supply room	Fill in / Edit Forms

Monday, March 25, 2002 Page 1 of 1

Figure 8.20: Report showing all work situations for a specific set of constraints.

Although in this example there is only one other *work situation*, which has exactly the same set of *constraints*, the user interaction designer can generate reports based on the specific

constraint categories to retrieve information about projects that might only differ in a few *constraint attributes* of a specific category. Thus, *ICE-Tool* allows for a basic data analysis beyond the exact match of all *constraint attributes*. Chapter 9 presents more examples of *work situations* that could be matched by *ICE-Tool* and a discussion of the distinguishing *constraints* of these *work situations*.

8.4. Entering / Retrieving User Feedback

To retrieve results during the report generation, *ICE-Tool* needs information about previously implemented user interfaces. Thus, after an implementation is done and field-tests have been conducted, the user feedback should be entered into the system. To illustrate the usefulness of the system, I entered as much user feedback as I could retrieve from the projects that I discussed in Chapter 7. Figures 20 and 21 show the user interface input dialog and the implementation notes dialog that support storing information about the selected user interface mechanisms for previous projects.

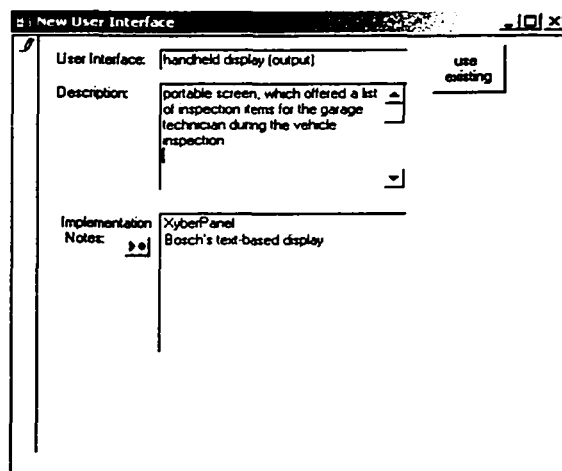


Figure 8.21: User interface input dialog

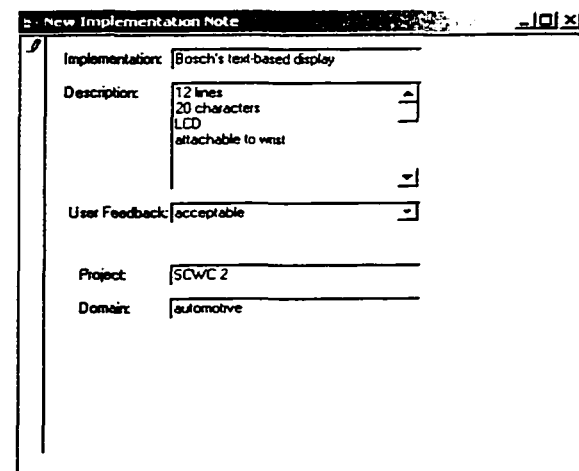


Figure 8.22: User interface implementation note dialog

The user interface implementation notes in *ICE-Tool* are on a high level because I could not retrieve more detailed information from the project reports found in the literature. However, the project, domain and user feedback information give a sufficient impression about the usefulness of the interface for the specific *work situation*. To retrieve information about user interface implementations of a specific *work situation*, the user has to select the *work location* and *work activity* and then selected a user interface from the corresponding list box as shown in Figure 8.22. After the user selects the kind of interface component, another dialog offers the actual implementation notes for a specific implementation or design decision.

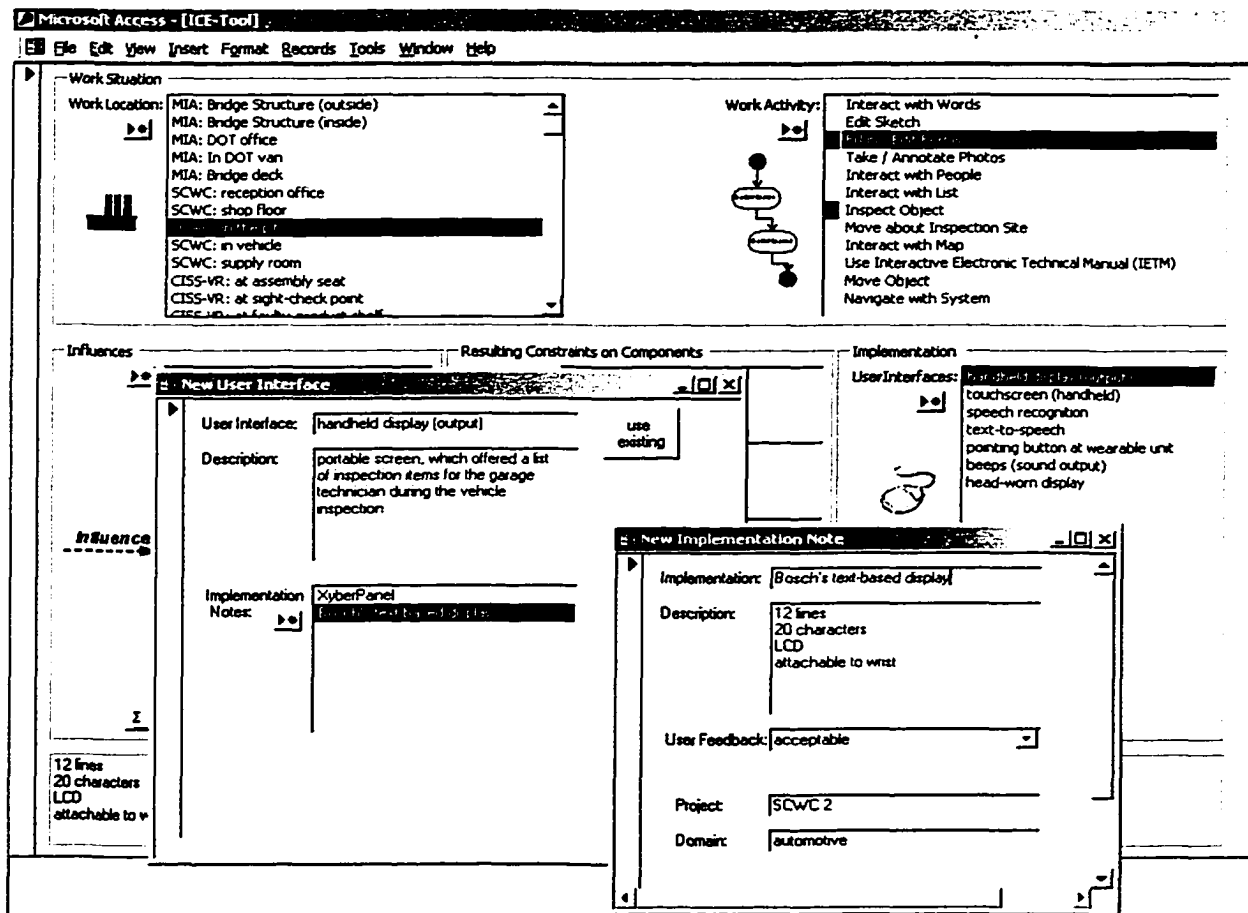


Figure 8.23: Dialogs showing user interface information for the selected work situation (filling in forms in the pit of a garage during the SCWC project).

9. The Interaction Constraints Model: Evaluation

In this chapter, I will evaluate the *Interaction Constraints Model* based on the proof-of-concept implementation, the “Interaction Constraints Evaluation Tool” (*ICE-Tool*), presented in Chapters 6 and 7, and the initial objectives and anticipated contributions presented in Chapter 3. Therefore, I first discuss the results of the proof-of-concept with *ICE-Tool*. Then, I summarize these objectives and intended contributions, followed by a discussion on whether or not the *Interaction Constraints Model* fulfills the objectives and delivers the intended contributions.

9.1. Discussion of Proof-of-Concept Results in ICE-Tool

The implementation of the *Interaction Constraints Model* in *ICE-Tool* allowed for evaluating the concept of matching different *work situations* based on the *constraints* caused by the combination of *work location* and *work activity*. The following sections discuss some examples that demonstrate this concept. The first section presents some anticipated results that an experienced designer might have obtained without using *ICE-Tool*. This demonstrated that the system works as expected. The second section presents some unexpected results from comparing *work situations* that came from the 15 projects discussed in Chapters 7 and 8. Finally, Section 9.1.3 presents some further examples that seemed to be challenging situations from different domains.

9.1.1. Discussion of Anticipated Results

While testing and evaluating *ICE-Tool*, I found several matches between *work situations* that could have been anticipated by an experienced designer of *m/w-CAE Systems*. These situations can be seen as reference situations that show the applicability of the concept of matching situations based on the *constraints* that occur at operation time.

Example 1: The first example covers situations in which the user of the mobile or wearable computer is situated in some kind of office environment to upload or post-process the collected data or to print a final inspection report or receipt for a customer. Since office environments usually are very similar and dealing with lists and forms means mostly dealing with text-based and table-based data, it is not surprising to see four of these situations matching in their *constraint patterns*.

Appendix B contains the *constraint pattern* that led to the matches listed Table 9.1.

Work Location	Work Activity
SCWC: reception office	Print safety inspection report
ICMMS: in construction site office	Upload progress data
CISS-VR: in office	Report time delay for assembly line
MobileDCT: in landfill office	Select route for monitoring

Table 9.1: Anticipated results for tasks performed in office environments, handling text-based and table-based data.

Discussion: Office environments differ from the rest of the *work locations*, mainly in terms of noise, cleanliness and roughness and often support activities related to handling list-based or form-based data. Thus, these four *work situations* build a distinct group within the database. Some *work situations* are similar to these four, such as a bridge inspection task performed at a bridge abutment, which only differs in the cleanliness and roughness.

Example 2: The second example covers inspection tasks that are performed in an outside location. Here, too, it is easy to anticipate matching *constraints*. Both situations deal with inspection-related tasks at outside locations, near machinery.

Appendix B contains the constraint pattern that led to the matches listed Table 9.2.

Work Location	Work Activity
ICMMS: in excavation area	Check progress
VuMan: vehicle (outside / outdoors)	Inspect vehicle

Table 9.2: Anticipated results for tasks performed at an outside inspection location, during observation / inspection processes.

Discussion: In this case, the distinguishing *constraints* for the two *work situations* result from the fact that both locations are outside in sunlight, with noisy machinery close-by, low cleanliness due to the construction site (ICMMS) or vehicle oil (VuMan), respectively, and the rough conditions under which the devices are used for the inspection task. Furthermore, in both situations, the user has one hand blocked for the inspection task with a tool or flashlight. Thus, the task *constraints* include the tool and the *user constraints* include the fact that holding this tool blocks the use of one hand of the user.

These two examples illustrate that *ICE-Tool* is capable of supporting assumptions made on the similarity of *work situations*. Thus, it is not only a tool to find similar *work situations*, but also to compare different *work situations* with respect to the applicability of user interfaces in these situations.

9.1.2. Discussion of Unanticipated Results within the Data Set

The following examples show matches of *work situations* that could not easily be anticipated, but can be explained by looking at the distinguishing *constraint attributes*. These examples demonstrate that the set of *constraints* for each *work situation* are comparable even though the applications and domains are different.

Example 3: The third example shows two *work situations* that were matched by *ICE-Tool* that cannot be related at first. The first one represents a task in a garage where a technician checks the spare parts from the supply room needed for a repair job

(SCWC); the second one describes a closed staircase in which a construction manager generates a daily report (ADL, see Chapter 8).

Appendix B contains the constraint pattern that led to the matches listed Table 9.3.

Work Location	Work Activity
SCWC: supply room	Collect parts for replacement
ADL: staircase inside	Document status of stairs

Table 9.3: Matching *work situations* for tasks performed at a garage respectively a construction site, during a selection and a documentation process.

Discussion: In these *work situations*, two different kinds of users have to browse through or enter data into lists while at *work locations*, which are fairly clean and silent. The low ambient light however, might restrict the visual input, such as lip reading and gesture recognition. Also, the visual cognition of the user is restricted in the staircase to prevent accidents while walking, and in the supply room by the fact that the user has to find the spare parts on the shelf.

Example 4: This example describes two matches of two *work situations*, each from two different domains: one from a maintenance application (Adtranz) and one from an emergency response support system (NOAH). The first two situations are located in a rather clean and silent environment, and deal with different inspection items: a public transport vehicle and a human being. The second pair of situations is situated in rather dark and noisy locations that involve dealing with other people. Appendix B contains the constraint pattern that led to the matches listed Table 9.4 and Table 9.5.

Work Location	Work Activity
Adtranz: train (inside / moving)	Browse through inspection items
NOAH: apartment	Enter diagnosis data

Table 9.4: Matching *work situations* for tasks performed at a moving vehicle and an apartment respectively, during a selection and a documentation process.

Work Location	Work Activity
Adtranz: tunnel	Contact help desk
NOAH: highway (at night)	Transmit patient data

Table 9.5: Matching *work situations* for tasks performed at a tunnel and on a highway at night respectively, during a communication process.

Discussion: Although the NOAH project is a medical application, it can still serve as an example for industrial applications. The matches derived from *ICE-Tool* show that both applications need to offer interfaces for different *work situations*. However, the user interaction is restricted by the same set of *constraints* and thus can be designed in a similar way. This example also illustrates that user interaction has to be defined based on each specific *work situation* and thus, an application might need a broader set of user interfaces of which the user can choose from in the specific situation.

Example 5: The fifth example shows another pair of *work situations*, which a designer might not have derived from literature. It illustrates once more that the same set of *constraints* can be found in different applications from different domains. Although the tasks of comparing a product catalog with the shelf content (CISS-VR) and retrieving tourist information in a bar or restaurant (Deep Map) seem to differ greatly, the interfaces to deliver the multimedia data and the *constraints* resulting from the *work location* match.

Appendix B contains the constraint pattern that led to the matches listed Table 9.6.

Work Location	Work Activity
CISS-VR: at faulty-product shelf	Use electronic spare part register
Deep Map: bar / restaurant	Browse multimedia tourist guide

Table 9.6: Matching *work situations* for tasks performed at a manufacturing shop floor and in a restaurant respectively, during the use of online documentation.

Discussion: In this example the match of the environmental *constraints* and the transfer from one domain to the other domain made it unlikely to imagine the match without *ICE-Tool*. The conditions in an industrial supply room are surely not the same as in a bar or restaurant, but they impact the design of an *m/w-CAE System* with the same set of *constraints*. Another finding about the Deep Map project is that using the system in a museum restricts the “Linguistic Ability” of the user and the “Audio Input” of the device. Thus, it matches in these categories to many other industrial applications. However, these restrictions do not result from the high ambient noise, which does not occur at a museum, but in the silence, which is expected from museum visitors, which does not allow for using speech input by the user.

9.1.3. Discussion of Further Examples

Each of the following examples presents *work situations* from applications that were not entered to the database before. These examples show that *ICE-Tool* can handle a great variety of *work situations* from different domains by entering the corresponding *constraint patterns*.

Example 6: This example refers to the design of a mobile navigation system that can guide firefighters through a smoke-filled building. Based on the *constraints* that can be expected for this *work situation*, *ICE-Tool* found a match with the OSCAR project in which the crane operator moves a load based on a graphical user interface that shows an interactive map with the location of the crane, the load and the supply boat respectively.

Appendix B contains the constraint pattern that led to the matches listed Table 9.7.

Work Location	Work Activity
OSCAR: crane cabin (severe conditions)	Lift and place load
Firefighter: smoke-filled building	Get guidance from navigation system

Table 9.7: Matching *work situations* for tasks performed in a crane cabin and in a smoke-filled building respectively, during the use of computer-based guidance.

Discussion: In this example, the *constraints* that distinguished these two *work situations* from the rest of the situations are mainly the *constraints* restricting the visual cognition of the user and the fact that in both situations, the user needs both hands. However, the device can still give visual feedback although the visual cognition is blocked, because the visibility within the crane cabin is not disturbed and also the firefighters can be supported by a head-worn display that might be integrated into the facemask.

Example 7: The last example compares the *constraint patterns* from an application in which an audio-only wearable computer [Vocollect 2001] supports a worker in a distribution center, to the *constraints* of a task from a landfill monitoring system (MobileDCT). The worker in the distribution center is fulfilling customer orders from shelves in different aisles. The landfill inspector uses a GPS system to get guidance to the next measurement point.

Appendix B contains that constraint pattern that led to the matches listed Table 9.8.

Work Location	Work Activity
MobileDCT: on uncovered landfill	Move about Inspection Site
Talkman: distribution center	Retrieve order picking information

Table 9.8: Matching *work situations* for tasks performed at a landfill and in a distribution center respectively, during the use of computer-based guidance.

Discussion: This example illustrates again that the *constraints* in the different categories can result from different *influences*. In the case of the landfill monitoring system, the visual output is blocked, because the inspector has to walk about an uncovered, cluttered landfill, which does not allow for checking any kind of display. During the order picking, the worker used an audio-only device, which implies a blocked visual output for the device. In both cases, the hands were blocked for the primary task (taking a measurement and handling groceries, respectively).

9.2. Initial Objectives and Anticipated Contributions

This section summarizes the initial objectives and the anticipated contributions introduced in Chapter 3. The objectives are separated into primary objectives, which are the basic objectives for the system developed in this research, and the secondary objectives, which are the demands on the data model behind the envisioned system that actually enable the primary objectives. Table 9.9 shows the primary objectives that illustrate the motivation for this research and the high-level goals at which the research was aimed.

O-01	The system should formally and generically describe the applicability of user interfaces with mobile and wearable computers for specific situations , especially with respect to speech interaction.
O-02	The system developed in this research should support system designers in deciding which mode of interaction is appropriate for a given activity in a given situation.
O-03	The system should speed up the design process in providing real-world data, use cases or patterns on how to implement given interaction modes, such as speech technologies.

Table 9.9: Primary objectives for a system implementing the *Interaction Constraints Model*

The primary objectives describe *what* functionality the system based on the *Interaction Constraints Model* is expected to offer. The secondary objectives represent *how* the model is expected to reach this functionality. Hence, the secondary objectives represent goals and high-level non-functional requirements that the underlying data model should fulfill to reach the primary objectives. Table 9.10 lists the secondary objectives.

O-04	The system has to be domain-independent (domain-neutral).
O-05	The system has to be implementation-independent .
O-06	The system has to be applicable to multi-modal interfaces .
O-07	The underlying data model has to be generic enough to fit a significant set of collected data (requirements or constraints), but narrow enough so as not to be too fuzzy.
O-08	The underlying data model has to be understandable by domain experts but also machine-readable for future use in adaptive interfaces.

Table 9.10: Secondary objectives for the *Interaction Constraints Model* itself

Table 9.11 lists the anticipated contributions of this research. These are contributions that result from the developed model itself as well as the description of the approach I took.

C-01	The main contribution of this research will be a decision support framework that supports domain experts with little development experience (i.e. non-CS developers) in realizing mobile and wearable computer systems for their domain.
C-02	A formal model of the requirements and constraints of the different design factors for the design of user interaction (esp. speech interaction) with mobile and wearable computers will be developed. This description will facilitate the interaction design process with categorized and formalized requirements and constraints.
C-03	As “tasks” are one of the needed design factors, I will provide a generic description for categorizing tasks and their relationships to other design factors (device, environment, etc.) and a guideline on how to map these tasks to the formal description mentioned in contribution C-02.
C-04	This research provides a case-base of real-world examples and user feedback that has been collected during the development of the proof-of-concept implementation.
C-05	As mentioned in C-01, this generic model can be used as a basis for adaptable and adaptive interfaces in the future.

Table 9.11: Anticipated contributions of the *Interaction Constraints Model*

During the analysis and development phase of the *Interaction Constraints Model*, I drew some conclusions and made assumptions, which went into the modeling process. I listed these as “statements” in Chapters 4 and 5. Table 9.12 summarizes these statements.

S-01	The design of interaction between users and mobile and wearable computers involves several disciplines, and thus is an activity that requires a huge amount of training.
S-02	Even actions that are not considered interaction with the machine have to be added to the interaction model for interaction with mobile and wearable computers to reflect the fact that operating these computers is only a supplementary task and not the primary goal of the user.
S-03	The design of speech interaction can serve as one example for a multi-modal user interface design for use in mobile and wearable computing.
S-04	Deciding about the usage of speech (or any other interaction mode) to support a specific activity is a binary decision and thus depends more on the relationship of the influencing parameters than on their priorities or values.
S-05	Providing an interactive guideline based on patterns and illustrative examples, the latter in an implementation-independent form, will provide substantial guidance for interaction design.
S-06	Regarding the constraints at operation time and especially the relationship between these constraints is essential for providing the right interface at the right time.
S-07	Classifying the constraints into a 5-component constraints model provides a clear basis for setting up an Interaction Constraints Model.
S-08	Orienting the task analysis around the actual work tasks reflects the actual constraints implied by the task and the environment at operation time.

Table 9.12: Statements or assumptions made during the development of the *Interaction Constraints Model*

These assumptions proved to be useful and true during the research. I kept these assumptions as a guideline for my research and, using them, I was able to develop a proof-of-concept prototype that meets nearly all of the objectives, which will be discussed in Section 9.3.

9.3. Discussion of Objectives

In this section, I evaluate how the *Interaction Constraints Model* and the proof-of-concept implementation, which is based on real-world project data, reached the objectives that I set up at the beginning of this research.

9.3.1. Primary Objectives

With the *Interaction Constraints Model* one can **formally and generically describe the applicability of user interfaces with mobile and wearable computers for specific situations (O-01)**. The main aspect of this research was on formalizing the design requirements and constraints of user interaction with mobile and wearable computers. Based on the challenges the research teams faced in developing different generations of prototypes of wearable computer systems, I started to focus on investigating the constraints that limit the use of a specific user interface component for a specific situation. The challenges were the changing environments in which *m/w-CAE Systems* are used and the switch of the interaction with the computer from a primary task (such as writing a business letter at a desktop computer) to a secondary task that supports another primary task (such as providing inspection information to a bridge inspector). Thus, the investigation of different *work situations* composed of *work locations* and *work activities* was a useful way to formalize this process. Furthermore, the separation into five different *constraint categories (task, environment, user, device, and application)* in contrast to the three categories used for stationary use of computers (*user, device, and application*) reflected the aspect of the mobility. This separation into different constraint categories also allows for describing how one *constraint* can compensate for another *constraint*, such as a noise-canceling microphone could compensate for a *constraint* caused by ambient noise and thus make this *constraint* obsolete.

The system reached the first primary objective, to **support system designers (O-02)** in their decisions about user interaction for specific situations, as shown in the illustrative example in Chapter 8. I showed that a user interface that is applicable for the given *constraints* can be

chosen based on the constraints of a specific situation and based on a previously collected knowledge base or case-base. The system is not an automated decision support system, but it does aid systems designers to make better-informed decisions. This system is useful especially if a design team has not conducted any prior projects dealing with *m/w-CAE Systems* in industrial applications.

If enough data is entered in the system so that a match between the design problem at hand and a set of previous cases can be made, based on a meaningful set of constraint patterns, the use of the system will **speed up the design process (O-03)** significantly. This faster design process results mainly from the possibility to identify work situations that are similar to the situation at hand. In some cases, where the kind of user interface that would best support the user is easy to imagine, a system, such as ICE-Tool can help to point to previously conducted projects that the designer would not have thought of for seeking examples for the design. To reach a meaningful set of project data to which to compare the situation at hand, the system must be in use and filled with real-world project information. For the proof-of-concept, I entered data from 15 projects (see Chapters 7 and 8), which was enough to show the concept of the *Interaction Constraints Model*, but will not be sufficient to support a complete industrial implementation of any *m/w-CAE System*. Thus, either future research has to build up a more complete data collection so as to populate the case-base, or in the case of a company-internal implementation, each new project will have to be added so as to contribute to the case-base and increase the usability for future projects. Hence, I see the objective of speeding up the design process fulfilled only after some time for data collection about previous or existing projects has been invested in advance. However, the *Interaction Constraints Model* provides a basis for representing the cases in this case-base.

9.3.2. Secondary Objectives

I evaluated the theoretical model against the secondary objectives already in Section 5.10, which showed that the model could fulfill them. The following presents a further evaluation

of the model and its proof-of-concept implementation in particular against these secondary objectives.

The first pair of secondary objectives required the system to be **domain-independent** (*O-04*) and **implementation-independent** (*O-05*). I decided to approach these two objectives by focusing on the theoretical model that underlies the system. In this way, I could ensure that the approach is not dependent on any implementation method or a development tool. In concentrating on *constraints* that impact the interaction with a mobile or wearable computer rather than on the *work situations* themselves, I could ensure the domain-independence of the model. *Work situations* are dependent on the application and thus on the domain, but *constraints* that occur in these situations are generic and thus exchangeable between applications and domains.

To make the system **applicable to multi-modal user interfaces** (*O-06*), I developed a model that maps user interface components to *constraints of work situations* and I did not restrict the model to a specific interaction mode or a specific number of different interaction modes. However, the usefulness of the system depends on the user interfaces that were entered in the system as successful or less successful implementations of previous projects. Thus, if interface combinations for a multimodal interface should be evaluated, all user interface components have to be present in the case-base to get meaningful decision support. However, this does not mean that the specific combination has to be implemented in the same way in one project before. It is enough to evaluate each user interface component based on the constraints that occur for the specific situations, no matter in which project and with which other interaction modes it was implemented before. Furthermore, in the current version of *ICE-Tool*, information about the kind of user interface component used is separated from information about the actual implementation of the component. For example, in the SCWC project we implemented two different handheld displays for the two generations of prototype system and thus, ICE-Tool shows two entries in the “implementation notes” listbox “handheld display”.

Objective *O-07* described the need to have the data model at the right level of detail to be **generic** and but **narrow** enough to let the system find enough similar *work situations* of previous

projects to support the design decision. To explore and reach this objective in a general way would have exceeded the scope of this research, because the final usefulness and the granularity of the captured and used data can only evolve over time. However, the granularity of the data set used in the proof-of-concept implementation was sufficient to show the concept of the *Interaction Constraints Model* and to prove its usefulness.

Finally, the last of the secondary objectives asked for a system that is **understandable by domain experts** and also **machine-readable** (O-08). In the illustrative example of Chapter 8, I showed that the system is easily usable, even without IT knowledge. The user has to elicit the *constraints*, enter them into the system, and start to evaluate the collected data. Thus, it is advantageous to have domain experts perform this process rather than software developers who do not know the domain at all. The machine-readability could not be shown in this proof-of-concept implementation, but I designed the data model in a way that could be easily transferred to a machine-readable format. In fact, if an automated query of the database were not acceptable, one fast approach to achieve this goal would be to parse the entity-relationship model that I used for the proof-of-concept into an XML schema definition and to enter the data into XML documents. This would result in a machine-readable format that is also platform-independent.

9.4. Contributions of this Research

In this section, I discuss the contributions of my research on the *Interaction Constraints Model*, which result from the developed theoretical model, the implementation of the proof-of-concept prototype, and the description of the approach or the technique of using the system itself, which can be adopted and transferred to other problem domains in the future.

The first contribution (C-01) is the **decision support framework**. The *Interaction Constraints Model* is not intended to be a decision support model itself, but as discussed in Chapter 4, it can serve as the “model base” for a decision support system. Together with a database that contains information about previously conducted projects, the *Interaction Constraints Model* may be used to build a system that can aid system designers in deciding about the right user interface for the right situation. I showed a proof-of-concept for this decision support

framework and I will discuss which issues would have to be investigated in more detail to make the system work beyond this research (see Chapter 10.2).

The above-mentioned model of user interaction will also serve as a **formal model of the requirements and constraints (C-02)** for the description of user interaction with *m/w-CAE Systems*. Besides the model itself, I introduced a technique that helps in identifying different user interface needs during task analysis, assigning tasks to different task categories and eliciting requirements and constraints for the use of specific user interfaces. The **generic description for categorizing tasks (C-03)** contributes to the interaction design process, since no formal description could be found in the literature. Most of the design guidelines focus on traditional user interfaces, such as graphical user interfaces, and assume a computer that is used on a desktop or in a kiosk setup, which means the user and the computer system are not moving. Furthermore, this research illustrated the transition of the interaction with the computer from a primary task (such as writing a business letter at a desktop computer) to a secondary task that supports another primary task (such as providing inspection information to a bridge inspector). This showed that the interaction paradigms and thus the design guidelines for this interaction have to be adapted. The *Interaction Constraints Model* is my approach to give guidance in the design process so that future project teams will not have to start from scratch for the interaction design, and thus may be able to save one or two field-test iterations or to focus on different aspects in the field-test.

Another contribution is the collection of **real-world examples and user feedback (C-04)** that I described in Chapter 7. Since most projects that develop *m/w-CAE Systems* remain in the prototyping phase, it is essential to build up this case-base of previously made experience. This research contributes to this need in two ways: first, I described the projects that went into the proof-of-concept in a general way, so as to give an overview of different approaches on using this technology, and second, I incorporated specific data on these projects into the database of the implementation, which allows for comparing different implementations of user interaction for specific situations. In the future, the system will provide a means to collect more real-world data in

an organized manner and to use this data as a case-base for supporting the design of future projects.

Finally, the system provides a formal description of the constraints that impact the interaction with a mobile or wearable device at operation time. This formal description can serve as a **basis for adaptable and adaptive interfaces (C-05)** in the future. This is only one small step towards adaptive or intelligent user interfaces, since there are many influences affecting this decision. But knowing and being able to describe the constraints in a specific situation allows for *automated decisions about the right interface for that specific situation*. Describing these constraints for specific work situations (locations and activities) helps the system to be context-aware, and thus allow for better-informed decisions about the right user interaction. Of course, to make such decisions, the user preferences, the system's performance, the connectivity and ergonomics of the system have to be investigated as well. These however, might be integrated into the *Interaction Constraints Model* in the future and thus make it a more solid and complete model. For that purpose, the research in these areas has to be included in the *Interaction Constraints Model* to enhance and extend the data structure of the model.

9.5. Summary of Evaluation

The evaluation in this chapter shows that the *Interaction Constraints Model* is a valid approach to formalize the user interface design for *m/w-CAE Systems* in industrial applications. This research proved that even domain experts, and not only system designers or human-computer interaction specialists, could apply the *Interaction Constraints Model* to improve the design process. Thus, we can develop a more advanced implementation that makes use of this model or an extended version of it to guide the interaction design process for *m/w-CAE Systems*. The concept of *the Interaction Constraints Model* can also be used in part "on the move," to enhance already established design processes by concepts that better reflect the mobility of mobile and wearable computer devices and the user interaction with these devices "on the move."

10. Conclusions

This chapter concludes this dissertation with a summary of my research and an outlook on how the outcome of this research can be used. Furthermore, the outlook illustrates necessary future research steps that will help to facilitate the interaction design with *m/w-CAE Systems* in industrial applications.

10.1. Summary of Research

In this research, I developed and illustrated an approach to improve the design process of mobile and wearable computer-aided engineering systems (*m/w-CAE Systems*). Domain experts, i.e., system designers whose background is in the domain in which the system will be used, rather than in system engineering or software development, need support during the user interaction design process. The *Interaction Constraints Model* proved to be a practical approach to meeting this need. Through the use of applications based on the *Interaction Constraints Model*, we can facilitate and speed up the design process for *m/w-CAE Systems* that are used in industrial environments.

The developed *Interaction Constraints Model* maps constraints of specific situations in which mobile IT support is needed to the user interface components that may be incorporated in the system design. Due to the nature of industrial applications, these situations mostly are work situations, i.e., situations in which the user of the mobile and wearable computers works at a specific location of the worksite and has to perform an actual job. This means that the user's interaction with the device is not only constrained by the physical location, but also by the activities that are supported by the device.

Defining work location and work activity as the two key identifiers for situations in which mobile IT support will be used, helped to identify the conditions for operating mobile and wearable computers in these different situations. The importance of location and activity evolved from the opportunity to establish IT support at the actual workplace through *m/w-CAE Systems*. The fact

that the computer support moved from a central location, such as the desktop or a kiosk-like computer, to “anywhere” on the worksite makes it inevitable during the design process to take into account the location of the mobile worker. The fact that mobile IT support helps to accomplish another activity - the actual job – requires that we view operating a mobile IT support only as a secondary task. Thus, this secondary task has to be unobtrusive with respect to the primary task and must not exhaust the cognitive and physiological capabilities of the worker, such as attention for the device, available hands for the device operation, or just willingness to use the device while performing another activity. Hence, each situation impacts the use of mobile and wearable computers with a set of constraints that result from the location in which the computer is used and the activity the computer supports.

Constraint patterns can help to identify the conditions of specific situations and thus describe these in an application-independent and domain-independent way. In focusing on constraint patterns – or sets of constraints – which affect the user interaction with the device and in mapping these constraints to usability information of user interfaces that were tested with these constraints, we can build up a generic description of constraint patterns that help to decide on the applicability of specific interfaces for certain situations. To define different constraint categories and to express the relationships between these categories, I investigated the changes of the device requirements and the changes in the usage environment that the mobility of *m/w-CAE Systems* brought. Before computers were mobile, the interaction was mainly influenced by three components: the computing device, the user and the application that was supported by the computer. Now, we face two more categories that have to be added: the environment in which the device is used and the task that the device supports. Thus, the design of mobile IT support is limited by constraints in respect to:

- the kind of the **task** to be performed;
- the **application**, for which the task is performed;
- the influences caused by the **environment** on the execution of the task;
- the **device** chosen as the supporting hardware platform; and
- the abilities and work patterns of the **user**.

The actual benefits for the design process result from the possibility to match an identified constraint pattern to a set of constraints that occurred in a work situation of a previously conducted project and thus to retrieve usability information for the different user interface components in that application. In this way, it is possible to retrieve information about previous projects that did not appear similar to the current project, and were thus not considered as examples for the current design.

The implementation of *ICE-Tool* based on the *Interaction Constraints Model* showed with real-world examples that matching work situations based on the constraints is a valid and workable approach. *ICE-Tool* demonstrated the concept and illustrated the necessary steps to identify work situations with similar work situations. To extend the *Interaction Constraints Model* and the *ICE-Tool* implementation will require future research steps that will be described in the next section.

10.2. Outlook and Future Research Steps

To use the *Interaction Constraints Model* in future decision support systems, some issues that occurred during the evaluation of the model have to be solved. Mainly, there are three aspects that should be addressed: 1) a refinement of the data model so as to support future use of data analysis methods; 2) an advanced version of the *ICE-Tool* implementation; and 3) the collection of more real-world data.

10.2.1. Refinement of the Data Model

One step for future research on the *Interaction Constraints Model* is the refinement of the data model that allows for a more precise description of *constraints*. This research shows that we can compare different *work situations* based on the *constraints* that occur in each situation. However, during the evaluation of the model, I could only compare *constraints* on a level of detail that was given in the projects described in the literature. A too detailed model in this evaluation

phase would have resulted in too many blank data entries for information that cannot be found in research publications and that maybe was not even collected in these projects. During the use of the proof-of-concept implementation of *ICE-Tool*, I identified some missing data, which would be useful to describe *constraint patterns of work situations* in the future. Examples for such data are additional attributes, such as information about the operating conditions for the device, and actual values and units for specific attributes, such as Centigrade for temperature, decibel for the noise level, or information about the needed bandwidth in Mbit/s for a network connection. Thus, the model can be refined in two ways: by including more attributes for each *constraint category* and by allowing more detailed values for each attribute. The following examples show how the refinement can be approached.

More attributes: While using *ICE-Tool*, I was able to sufficiently describe each *work situation* with the given *constraints attributes*. However, in some cases, I had information that could have described the situations in more detail if the model had a way to represent this information, maybe in an advanced level of detail, which could be assessed after the top level matching of *work situations* generated too many results. For example, a separation of the “roughness” attribute into “operating temperature”, “sealing”, and “drop specification” in the *environment constraints*, or a more detailed description of the needs for “data access” in terms of needed bandwidth, the latency of the data transfer, or the refresh rate for the data would be helpful. Another example of a separation into different attributes is the information about “interaction with other people”. During the evaluation of *ICE-Tool*, this attribute linked two situations in which the user had to interact with people either face-to-face or via an audio communication, which certainly requires different interfaces. However, these examples imply that we can collect this data for future projects, which would further imply a *constraints elicitation* process according to data model of the *Interaction Constraints Model*.

More details for each attribute: For example the attributes that capture the ambient noise level of a specific *work situation* should allow for entering values in decibel and the light conditions attribute should be measured and entered in Lux. With these given values, we could

go beyond high-level decisions, such as “ambient noise too loud” and “ambient noise level acceptable”.

With more precise *constraint* descriptions that can be captured and documented, it will be possible to implement more detailed data analysis methods on the given data. Based on that, we can set operating thresholds for the components of the anticipated device, such as a maximum decibel level for operating a noise-canceling microphone, to more exactly predict if speech input is possible in a specific situation. In this way, we can also vary the different components to fit the device to the constraints that will occur at the usage situations. To come back to the “tool analogy” from Chapter 1, the model will help the task-specific design (single tool) as well as the design of devices that offer all applicable interfaces from which the user can choose (Swiss army knife), and will help to set up a model that can support future adaptive user interfaces.

For using the *Interaction Constraints Model* in adaptive interfaces, the device will have a noise sensor and enable or disable the specific interface based on the measured values, similar to the automatic backlight adjustment of current PDA models. For adaptive user interfaces, another important refinement would be to add a user model, which not only captures the physiological and cognitive constraints that impact the user, but also the personal preferences of a user and the user’s usage history, which was not in the scope of this research.

10.2.2. Collection of More Real-World Data

The data collection that I described in Chapter 7 was sufficient for a proof-of-concept with the current implementation of *ICE-Tool*, but is not broad and deep enough to run various types of queries on *constraints patterns*. For that reason, more projects have to be included in this case-base after the data model was refined as discussed in the previous section. This will be a great effort and might involve the contribution of different research teams to get a sufficient case-base. On the other hand, the system can also be set up as a more specific implementation that captures only information about one kind of applications or user interface components and thus the data source can be kept more concise and smaller.

Another issue that occurred was how outdated information about older projects could be handled. In the current version of *ICE-Tool*, it is possible to enter several implementation notes for a specific interface component, such as different handheld displays or different speech recognition engines used for a specific project. Adding version information to the implemented user interface components could extend this approach. Thus, the collected data could be queried for the most advanced implementations for a specific interface. Furthermore, the system could not only collect data about projects and work situations, but also data about user interface components. These descriptions of the user interface components could be given by the manufacturers and serve as a catalog of user interfaces and their applicability for specific *constraints*. This means that similar to operating conditions given in device specification, developers of user interfaces could give a worst-case usage scenario, based on operation time *constraints*. As such, the system could serve as a user interface design aid: New user interface concepts could be entered into the system with several *constraints patterns*, for which the interface was designed. In that way, the newly created interface can be tested against several *work situations* of previous projects in which the interfaces are designed to work. Hence, the *Interaction Constraints Model* cannot only aid the decision on which interface to choose, but also on how to create new interfaces that improve the interaction with future *m/w-CAE-Systems* in industrial applications.

10.2.3. Future ICE-Tool Implementation

The *Interaction Constraints Model* can be used as the model base of future *ICE-Tool* implementations to support interaction design with *m/w-CAE Systems*. For these future implementations, I recommend to transfer the model into an XML data structure, which will provide for a standardized and platform-independent format. In this way, the sets of collected data from previous projects could be stored on a central server and be accessed by web clients through the intranet or Internet between different design teams using different development environments on different platforms. Also, the automated decision about the best user interface

for a specific *work situation* could be made based on *constraints* gathered at operation time instead of design time, i.e., during the actual use of the *m/w-CAE System*. In this case, the information could be provided by the different components that are involved in a *work situation* by individual XML descriptions, such as a user file, task definition, a bridge description or a device configuration file. These files could then be processed into a set of *constraints* for the specific *work situation* and build a decision basis for determining the best user interface. Thus the user interaction could become adaptive and context-aware based on the *Interaction Constraints Model*.

Using and evaluating *ICE-Tool* helped to detect some opportunities for improvement for future addition to the implementation:

Copy & Paste: Future versions of *ICE-Tool* should offer functionality to copy specific entries or whole *constraint patterns* to re-use them in other situations. For example if different situations for a specific *work location* have to be entered, many *constraints* can be kept the same. Thus, it would speed up the data entry significantly, if *ICE-Tool* would offer an intelligent “copy & paste” functionality.

Entry wizard: Another improvement would be a data entry wizard, which would guide the user through the process of entering *constraint* information. The current version of *ICE-Tool* assumes that the user knows the entry procedure and does not verify if information is missing or ambiguous. This wizard could also help to establish a standardized *constraints* elicitation process that ensures that each work situation is captured in the same level of detail and completeness.

Variation of constraints: To help in analyzing the data, especially with respect to the interface design based on the *Interaction Constraints Model*, options to vary the values of certain *constraint* attributes would be helpful. For examples sliders, which vary the ambient noise level and lighting levels, would help to more quickly change the conditions entered for a specific *work situation*. This would be another step to a more variable model that could adapt to actually measured values of some *constraint attributes*.

Prediction capability: Another functionality, which could help improving the analysis and design process, would be a prediction capability of the system that could suggest *constraint attributes* based on previously entered work situations. This would require the system to capture the essence of the entered *work location* and *work activity* and to transfer these into other location and activity combinations. For example, if the system knows the constraints of a list-based data entry activity at low light, high noise situation, it could predict the constraints for dealing with graphical or audio data at the same location. However, to implement such algorithms, the system has first to be established and filled with statistically reliable data.

Finally, there are two usage scenarios for *ICE-Tool* that may be considered to improve the management of interaction constraints: *ICE-Tool* can be used as a form of knowledge management system to keep project data collected for a company, so the different design considerations can be retrieved even if other documentation means did not cover *constraints* very well and the responsible system designers are no longer available. The second usage scenario links back to the citation of the beginning: we should consider using an *m/w-CAE System* to capture the *constraints* directly at the *work locations* and during the task analysis in digital form and thus “*tame the monster with a piece of itself*”.

Appendixes

Appendix A: Additional Reports for Illustrative Example

The following is a table-based summary of all *work situations* that the generated reports based on single *constraint categories* contained for the *work situation* "Interact with List" in an "Inside staircase" of the *Automated Daily Log* project (see Chapter 8).

Task Constraints (5 matching work situations)

Work Location	Work Activity
Adtranz: Train (inside / moving)	Inspect Object
MIA: Bridge Structure (outside)	Interact with List
MIA: Bridge Structure (inside)	Edit Sketch
SCWC: supply room	Fill in / Edit Forms
ADL: Staircase inside	Interact with List

Environment Constraints (7 matching work situations)

Work Location	Work Activity
MIA: In DOT van	Interact with People
ICMMS: on finished floor	Take / Annotate Photos
ICMMS: on finished floor	Interact with List
MIA: In DOT van	Fill in / Edit Forms
SCWC: supply room	Fill in / Edit Forms
MobileDCT: in landfill vehicle	Interact with List
ADL: Staircase inside	Interact with List

User Constraints (14 matching work situations)

Work Location	Work Activity
Stars: Power Plant (outside)	Use Interactive Electronic Technical Manual (IETM)
MIA: Bridge Structure (outside)	Interact with List
SCWC: shop floor	Move about Inspection Site
VuMan: Vehicle (outside / outdoors)	Inspect Object
Navigator: Aircraft (outside, outdoors)	Move about Inspection Site
Adtranz: Tunnel	Interact with People
ADL: Staircase inside	Interact with List
ICMMS: in excavation area	Fill in / Edit Forms
SCWC: supply room	Fill in / Edit Forms
ICMMS: in excavation area	Inspect Object
MIA: Bridge Structure (inside)	Interact with List
MobileDCT: on uncovered landfill	Interact with People
MobileDCT: on covered landfill	Inspect Object
MobileDCT: on covered landfill	Interact with Map

Device Constraints (4 matching work situations)

Work Location	Work Activity
Adtranz: Train (inside / moving)	Inspect Object
Adtranz: Tunnel	Interact with People
ADL: Staircase inside	Interact with List
SCWC: supply room	Fill in / Edit Forms

Application Constraints (34 matching work situations)

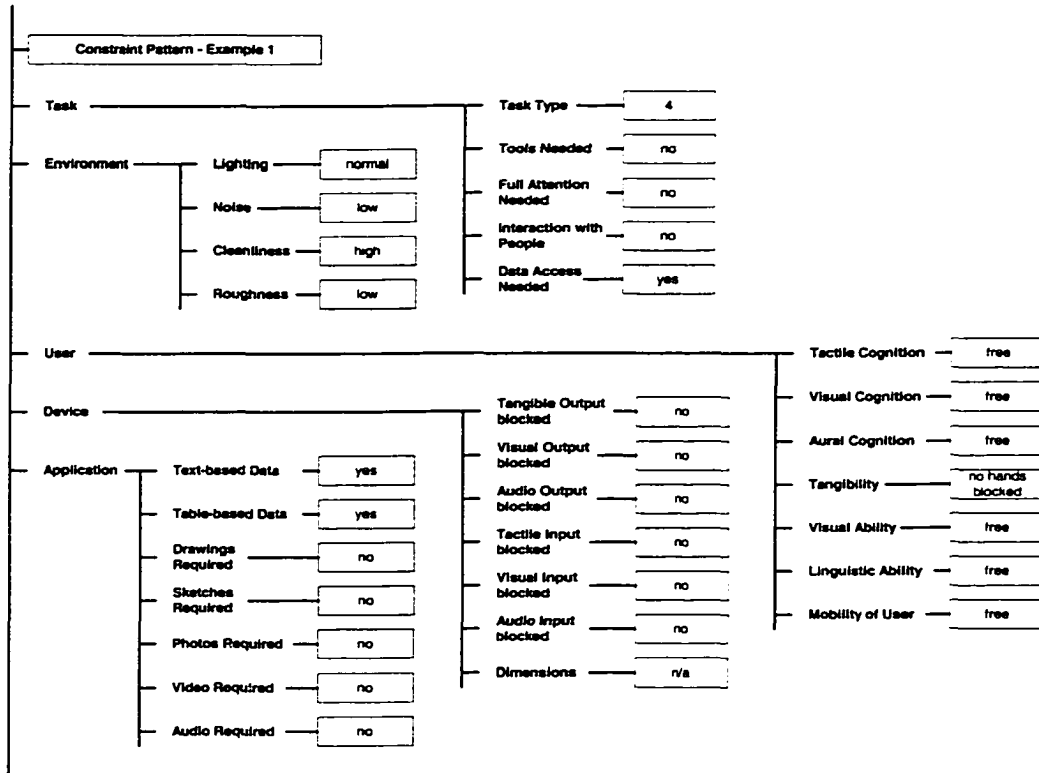
Work Location	Work Activity
Painter: Roof	Fill in / Edit Forms
MobileDCT: in landfill vehicle	Interact with List
Stars: Helium Flashing System	Inspect Object
ICMMS: in construction site office	Interact with List
VuMan: Shop Floor	Fill in / Edit Forms
ADL: Staircase inside	Interact with List
Navigator: Scaffolding (outside aircraft)	Fill in / Edit Forms
ICMMS: on finished floor	Interact with List
ICMMS: in tunnel / basement	Interact with List
ICMMS: in excavation area	Fill in / Edit Forms
MobileDCT: on covered landfill	Inspect Object

Work Location	Work Activity
VuMan: Pit	Interact with List
MobileDCT: in landfill office	Fill in / Edit Forms
CISS-VR: in office	Fill in / Edit Forms
CISS-VR: at sight-check point	Interact with List
CISS-VR: at assembly seat	Interact with List
MIA: In DOT van	Fill in / Edit Forms
SCWC: reception office	Fill in / Edit Forms
MIA: DOT office	Fill in / Edit Forms
MIA: Bridge Structure (inside)	Interact with List
MobileDCT: at reading spot	Fill in / Edit Forms
Winspect: Steel Mill	Fill in / Edit Forms
DHH: Tunnel / Basement	Interact with List
Adtranz: Train (inside / moving)	Fill in / Edit Forms
NOAH: Highway (heavy traffic)	Fill in / Edit Forms
NOAH: Highway (at night)	Fill in / Edit Forms
NOAH: Vehicle Wreckage	Fill in / Edit Forms
NOAH: Appartment	Interact with List
NOAH: Sports Arena	Interact with List
SCWC: in vehicle	Fill in / Edit Forms
SCWC: in the pit	Fill in / Edit Forms
Deep Map: Hotel	Fill in / Edit Forms
SCWC: supply room	Fill in / Edit Forms
Winspect: Crane	Fill in / Edit Forms

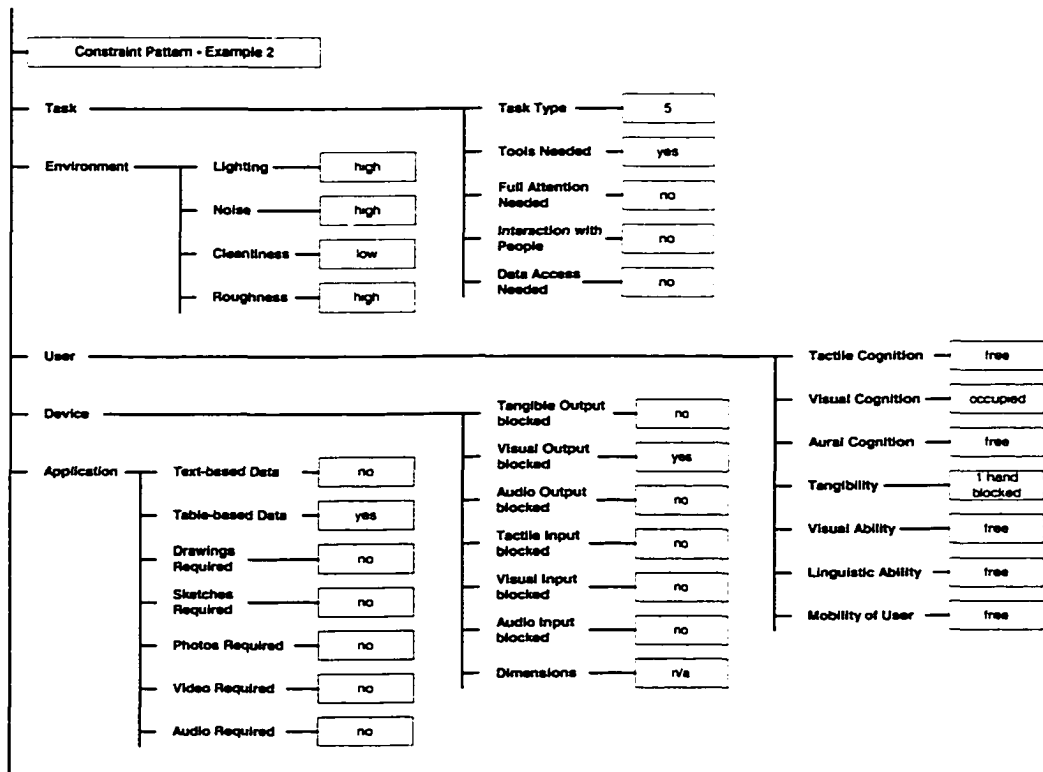
Appendix B: Constraint Patterns of ICE-Tool Examples

The following are graphical representations for the *constraint patterns* that led to matching the different *work situation* in Chapter 9.1.

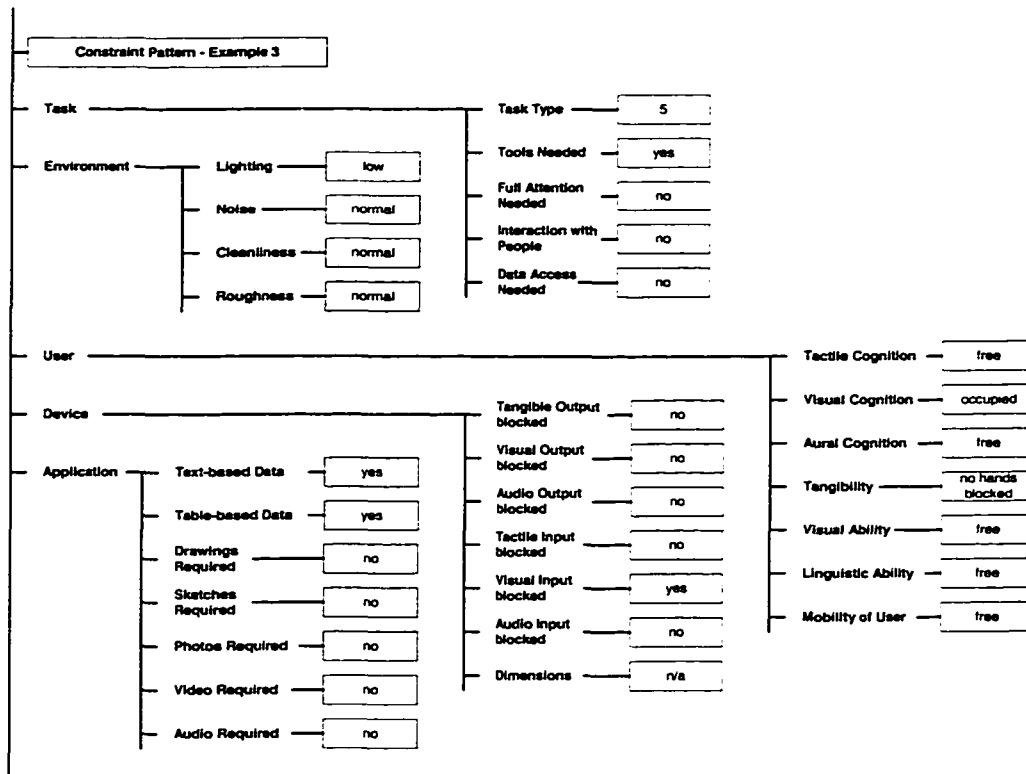
Constraint Pattern of Example 1:



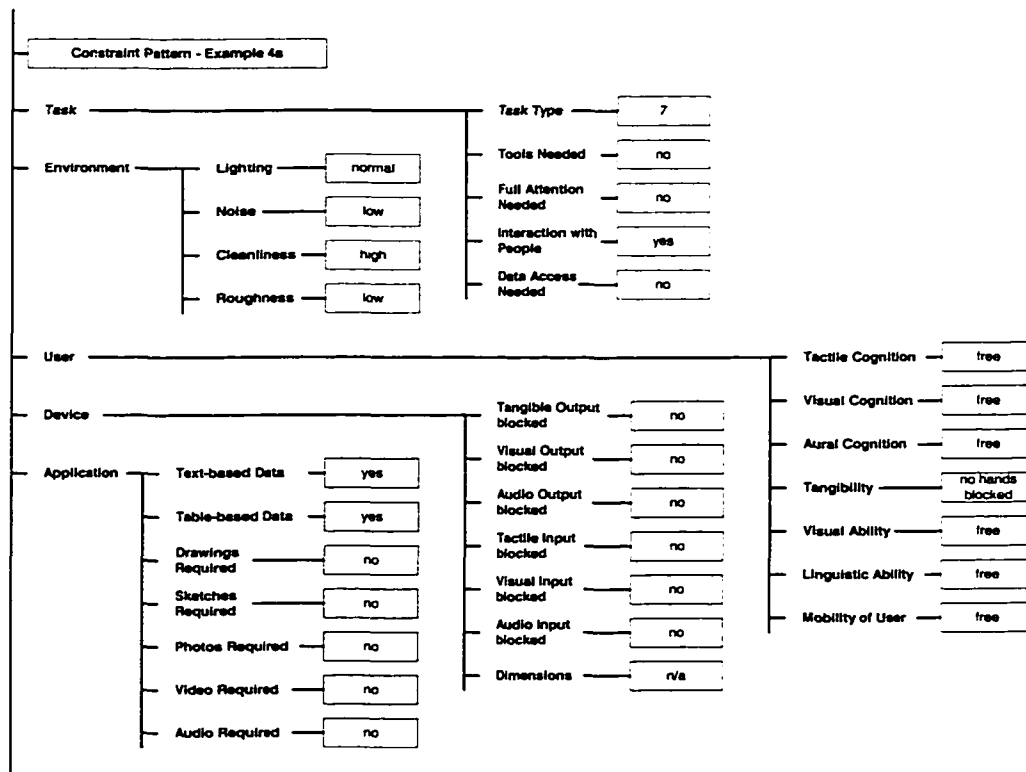
Constraint Pattern of Example 2:



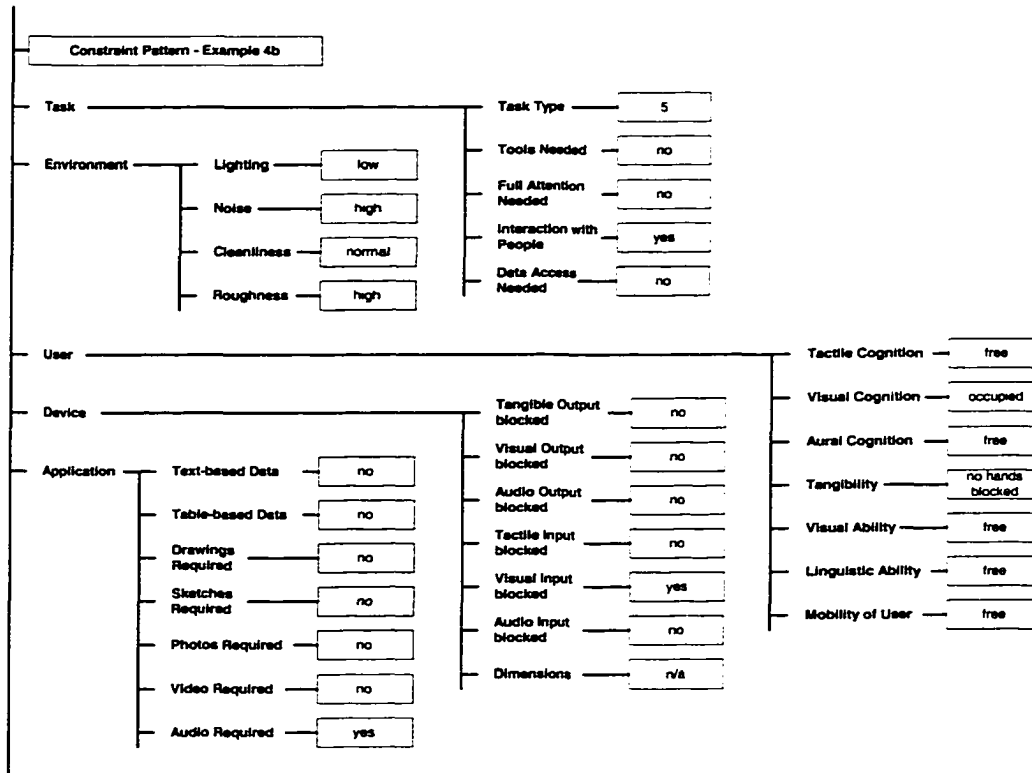
Constraint Pattern of Example 3 (same as ADL example from Chapter 8):



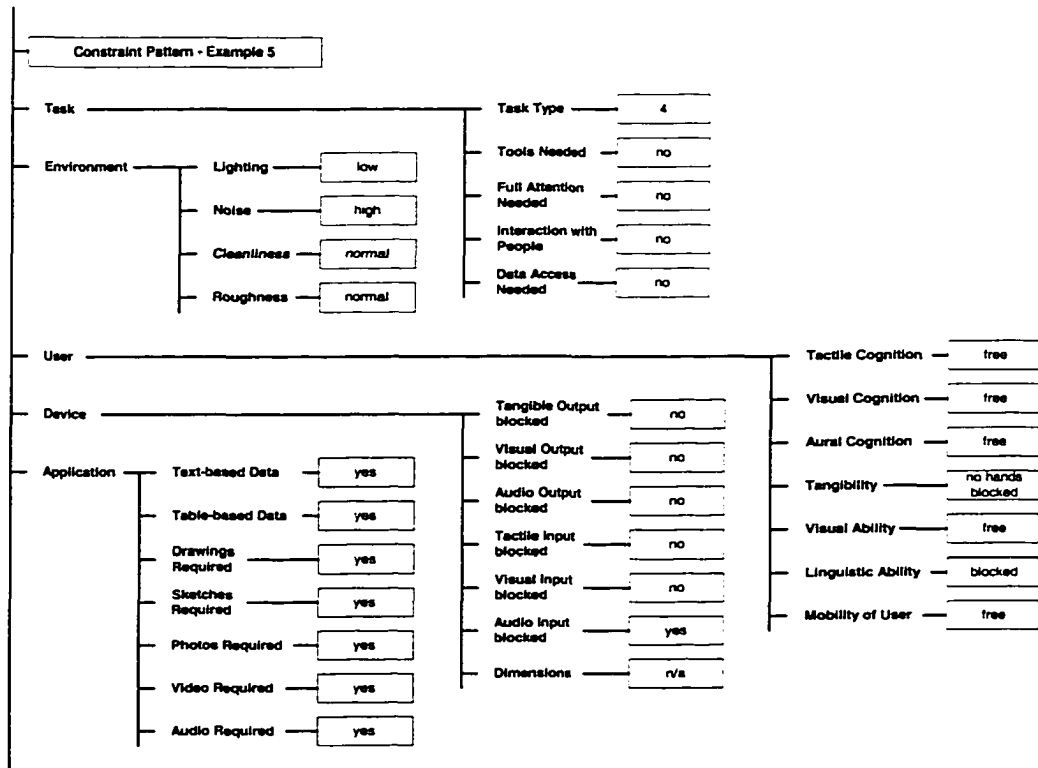
Constraint Pattern of Example 4a:



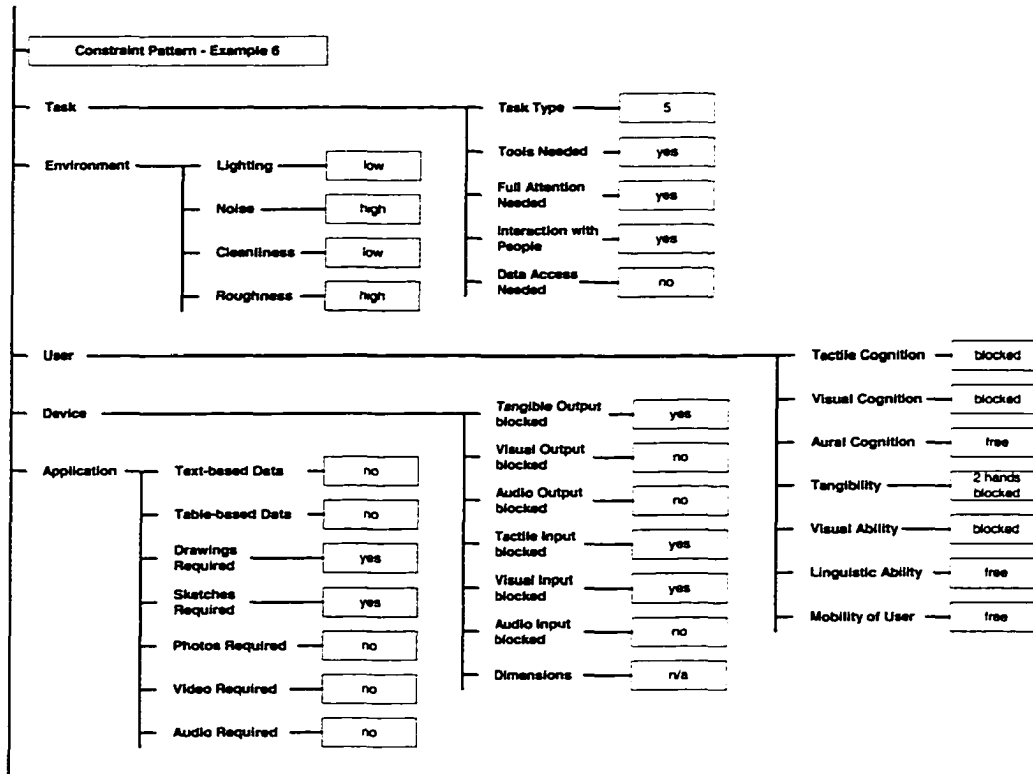
Constraint Pattern of Example 4b:



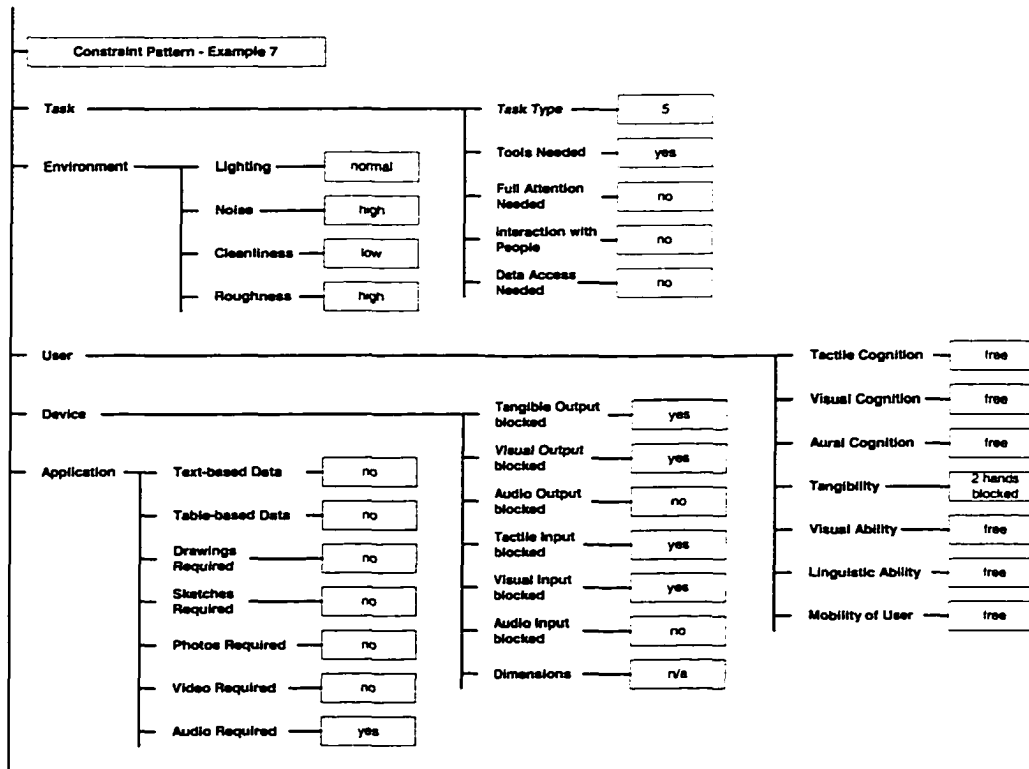
Constraint Pattern of Example 5:



Constraint Pattern of Example 6:



Constraint Pattern of Example 7:



References

- [aecXML 1999] The XML Cover Pages. 1999 - *aecXML Working Group - Architecture, Engineering and Construction*.
<<http://xml.coverpages.org/aecXML.html>> (retrieved 19 December 2001).
- [Anind 1999] Anind K. Dey., Daniel Salber, Gregory D. Abowd. 1999. *A Context-Based Infrastructure for Smart Environments*. Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments, Dublin, Ireland, December 1999.
- [Azuma 1997] Azuma, Ronald T. 1997. *A Survey of Augmented Reality*. In *Presence: Teleoperators and Virtual Environments* 6, 4 (August 1997), 355-385.
- [Azuma 2001] Ronald Azuma, Yohan Baillet, Reinhold Behringer, Steven Feiner, Simon Julier, Blair MacIntyre. 2001. *Recent Advances in Augmented Reality*. *IEEE Computer Graphics and Applications* 21, 6 (Nov/Dec 2001), pp. 34-47.
- [Baber 1999] Baber, Chris, David Haniff, Sandra Woolley. 1999. *Contrasting paradigms for the development of wearable computers*. *IBM Systems Journal*, Vol. 38, No. 4 1999.
- [Balasubramanian 1998] Balasubramanian, Anitha, Julie Rodriguez, Gautam Kharkar, and Neeraj Bansal (editors). 1998. *Mobile Inspection Assistant (MIA) – Final Report*. Course Project “Wearable Computers”, Spring 1998, Carnegie Mellon University, Pittsburgh, PA, USA:
<<http://www.ce.cmu.edu/~wearables/#projects>> (retrieved 06 February 2002)
- [Balzert 1996] Balzert, Helmut. 1996. *Lehrbuch der Software-Technik: Software-Entwicklung*. Heidelberg, Germany: Spektrum, Akademischer Verlag.
- [Barfield 2001] Barfield Woodrow, Kevin Baird, John Skewchuk, George Ioannou. 2001. *Applications of Wearable Computers and Augmented Reality to Manufacturing*. In *Fundamentals of Wearable Computers and Augmented Reality*. Woodrow Barfield and Thomas Caudell (Editors). Lawrence Erlbaum Associates, Inc.: Mahwah, NJ, USA.

References

- [Bass 1997] Bass, Len, Chris Kasabach, Richard Martin, Dan Siewiorek, Asim Smailagic, and John Stivoric. 1997. *The design of a wearable computer*. In Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems, volume 1 of PAPERS: Input & Output in the Future, pages 139-146, 1997
- [Bass 2000] Bass, Len. 2000. *Interaction Technologies: Beyond the Desktop; Chapter in User Interfaces for All*. Lawrence Erlbaum Associates, pp. 81-95.
- [Bass 2001] Bass, Len, Dan Siewiorek, Malcom Bauer, Randy Casciola, Chris Kasabach, Richard Martin, Jane Siegel, Asim Smailagic, John Stivoric. 2001. *Constructing Wearable Computers for Maintenance Applications*. In Fundamentals of Wearable Computers and Augmented Reality. Woodrow Barfield and Thomas Caudell (Editors). Lawrence Erlbaum Associates, Inc.: Mahwah, NJ, USA.
- [Beaudouin-Lafon 2000] Beaudouin-Lafon, Michel. 2000. *Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces*. Proceedings CHI 2000, pp.446-453.
- [Bennington 1999] Bennington, Ben. 1999. *A Model for Networking of Field Support and Maintenance Operations for Daimler Benz*. Research report. <<http://www.ini.cmu.edu/WIRELESS/Daimler/>> (retrieved 17 January 2002).
- [Bernsen 1997] Bernsen, Niels Ole, Hans Dybbkjær, Laila Dybbkjaer. 1997. *What should your computer say?* IEEE Computer, Vol. 10, No. 12, December 1997.
- [Bernsen 1999] Bernsen, N. O. and Luz, S. 1999. *SMALTO: Advising interface designers on the use of speech in multimodal systems*. In Ostermann, J., Ray Liu, K. J., Sørensen, J. A., Deprettere, E. and Kleijn, W. B. (Eds.): Proceedings of the Third Institute of Electrical and Electronics Engineers (IEEE) Workshop on Multimedia Signal Processing, Elsinore, Denmark, September, 1999. IEEE, Piscataway, NJ, USA; 489-494.
- [Bhargava 1999] Bhargava, Hemant K.; Sridhar, Suresh; Herrick, Craig. 1999. *Beyond Spreadsheets: Tools for Building Decision Support Systems*. IEEE Computer, Vol. 12, No. 3, March 1999. pp.31-39
- [Billinghurst 1998] Billinghurst, M., J. Bowskill; Nick Dyer; Jason Morphett. 1998. *An Evaluation of Wearable Information Spaces*. Virtual Reality Annual International Symposium, Atlanta, GA, USA 1998.

References

- [Billinghurst 1999] Billinghurst, Mark; Thad Starner. 1999. *Wearable Devices, New Ways to Manage Information*. IEEE, Computer, Vol. 32, No. 1, January 1999. pp. 57-64.
- [Booch 1998] Booch, Grady, James Rumbaugh, Ivar Jacobson. 1998. *The Unified Modeling Language User Guide*. Reading, MA, USA: Addison Wesley Longman Inc.
- [Boronowsky 2001] Boronowsky, Michael. 2001. *Winspect: A Case Study for Wearable Computing-Supported Inspection Tasks*. Proceedings of The Fifth International Symposium on Wearable Computers, Zürich, Switzerland, 8-9 October 2001.
- [Borrmann 2001] Borrmann, Alf, Stefan Komnick, Gunnar Landgrebe, Jan Matèrne Manfred Rätzmann, Jörg Sauer. 2001. *Rational Rose und UML – Anleitung zum Praxiseinsatz*. 1st Edition. Bonn, Germany: Galileo Press GmbH.
- [Bradley 1998] Bradley, Neil. 1998. *The XML companion*. Harlow, England: Reading, Mass.: Addison-Wesley.
- [Bredenfeld 2000] Bredenfeld, Ansgar, Edmund Ihler, Oliver Vogel. 2000. *GENVIS - Model-Based Generation of Data Visualizers*. Technology of Object-Oriented Languages and Systems (TOOLS 33), St. Malo, France.
- [Brewer 1998] Brewer, E., R. H. Katz, E. Amir, H. Balakrishnan, Y. Chawathe, A. Fox, S. Gribble, T. Hodes, G. Nguyen, V. Padmanabhan, M. Stemm, S. Seshan, and T. Henderson. 1998. *A Network Architecture for Heterogeneous Mobile Computing*. IEEE Personal Communications Magazine, Vol 5, No 5, October 1998.
- [Bruegge 2000] Bruegge, Bernd and Allen H. Dutoit. 2000. *Object-oriented Software Engineering*. Prentice-Hall, UK.
- [Buergy 2000] Bürgy, Christian. 2000. *Speech-Controlled Wearable Computer - A mobile system supporting inspections in garages - final project report*. Carnegie Mellon University, Robert Bosch Corporation, Pittsburgh, USA, March 31, 2000: <<http://www.ce.cmu.edu/~wearables/#projects>> (retrieved 04 February 2002)
- [Buergy 2000a] Bürgy, Christian. 2000. *Speech-Controlled 3D-Objects*. <<http://www.ce.cmu.edu/~buergy/projects/index.html#3D-Objects>> (retrieved December 2001)
- [Buergy 2001] Bürgy, Christian, James H. Garrett Jr., Markus Klausner, Jürgen Anlauf, Günter Nobis. 2001. *Speech-Controlled Wearable Computers for Automotive Shop Workers*. SAE World Congress, Detroit, MI, USA.

References

- [Buergy 2001a] Buergy, Christian and James H. Garrett, Jr. 2001. *Wearable Computer: Der Entwurf mobiler IT-Systeme zur Unterstuetzung von Arbeitsprozessen im Bauwesen*. Fortschritt-Berichte VDI - Junge Wissenschaftler forschen, Reihe 4, Nr. 169, Forum Bauinformatik 2001, Muenchen, Germany. VDI Verlag GmbH, Duesseldorf, Germany. pp 127-134.
- [Buergy 2002] Būrgy, Christian and James H. Garrett, Jr. 2002. *Wearable Computers: An Interface between Humans and Smart Infrastructure Systems*. Bauen mit Computern 2002, Bonn, Germany. VDI Verlag GmbH, Duesseldorf,
- [Calhoun 1998] Calhoun, Gloria L. and Grant R. McMillan. 1998. *Hands-free Input Devices for Wearable Computers*. Fourth Symposium on Human Interaction with Complex Systems, Dayton, OH, USA.
- [Caple 2001] George Caple, David Haines, Dave Lamont et. al. 2001. *Requirements Categorization*. Prepared by the Requirements Working Group of the International Council on Systems; Engineering, for information purposes only. <<http://www.incose.org/rwg/>> (retrieved 29 October 2001).
- [Chapin 1999] Chapin, J.K, Moxon, K.A., Markowitz, R.S. and Nicolelis, M.A.L. 1999. *Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex*. Nature Neurosci., 2:664-670.
- [CII 2002] CII: Construction Industry Institute. 2002. *Digital Hardhat - Based Tools for Team Collaboration* <<http://www.new-technologies.org/ECT/Other/digitalhat.htm>> (retrieved 24 February 2002).
- [CMU 1996] Carnegie Mellon University. 1996. *Boeing Wearable Computer Workshop Breakout Session Summary*. <<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/vuman/www/boeing/index.html>> (retrieved 20 March 2001).
- [CMU 1997] Carnegie Mellon University. 1997. *Navigator 2*. <<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/vuman/www/navigator.html>> (retrieved August 30, 2001).
- [Cohen 1994] Cohen, P. R., S. L. Oviatt. 1994. *The Role of Voice in Human-Machine Communication*. In Voice Communication Between Human and Machines, D.B. Roe and J. G. Wilpon (Eds), pp. 34-75.
- [Compaq 2002] Compaq. 2002. Homepage. <<http://www.compaq.com/>> (retrieved 19 February 2002).

References

- [Coors 1999] Coors, V. and U. Jasnoch. 1999. *Deep Map: A Virtual Tourist Guide in Heidelberg*. In Computer Graphik TOPICS 2/99, volume 11, pages 13-15.
- [Cress 1997] Cress, Jeffrey D., Lawrence J. Hettinger, and James A. Cunningham et al. 1997. *An Introduction of a Direct Vestibular Display into Virtual Environments*. IEEE Computer Graphics and Applications, November-December 1997 (Vol. 17, No. 6)
- [Damper 1993] R. I. Damper. 1993. *Speech as an Interface Medium: How can it best be Used?* In *Interactive Speech Technology: Human Factors Issues in the Application of Speech Input/Output to Computers* pp. 70. Taylor and Francis.
- [Davis 1993] Davis Alan, M. 1993. *Software Requirements: Objects, Functions and States*. Englewood Cliffs, N.J. USA: Prentice Hall.
- [Dell 2002] Dell. 2002. Homepage. <<http://www.dell.com>> (retrieved 19 February 2002).
- [Denecke 1997] Matthias Denecke and Alex Waibel. 1997 *Dialogue strategies guiding users to their communicative goals*. Proceedings of Eurospeech97: Rhodes, Greece.
- [Disc 1999] Disc. 1999. *Working Paper on Speech Functionality*. Niels Ole Bensen and Laila Dybækjær, Natural Interactive Systems Laboratory, Odense University, Denmark: <<http://www.disc2.dk/publications/deliverables/>> (retrieved 11 November 2000)
- [Druzdzel 2000] Druzdzel Marek J. and Roger R. Flynn. 2000. *Decision Support Systems*. In *Encyclopedia of Library and Information Science*, Vol. 67, Suppl. 30, pages 120-133, Allen Kent (ed.), Marcel Dekker, Inc., New York, USA.
- [Dutoit 2001] Dutoit, Allen H., Oliver Creighton, Gudrun Klinker, Rafael Kobylinski, Christoph Vilsmeier, Bernd Bruegge. 2001. *Architectural Issues in Mobile Augmented Reality Systems: A Prototyping Case Study*. Proceedings of 8th Asia-Pacific Software Engineering Conference (APSEC 2001), Macau, December 4-7, 2001.
- [ebXML 2001] Oasis and UN/CEFACT. 2001 <<http://www.ebxml.org/>> (retrieved 19 December 2001).
- [Elting 2002] Elting, Christian, Jan Zwickel, Rainer Malaka. 2002. *Device-Dependant Modality Selection for User-Interfaces - An Empirical Study*. International Conference on Intelligent User Interfaces (IUI 2002), January 13-16, 2002, San Francisco, CA, USA.

References

- [Espisito 1997] Espisito, C. *Wearable Computers: Field-Test Results and System Design Guidelines*. Proceedings Interact '97, Sydney, Australia, 1997.
- [Extreme 2002] Extreme Computing. 2002. *Finger Trackball Mouse*. <<http://www.extremecomputing.com/fin.html>> (retrieved 10 January 2002)
- [FieldWorker 2001] *Field Worker -The mobile edge in data collection software*. <<http://www.fieldworker.com>> (retrieved November 17, 2001)
- [Fowler 1997] Fowler, Martin. 1997. *Analysis Patterns: Reusable Object Models*. 5th printing. Menlo Park, CA USA: Addison Wesley Longman, Inc.
- [Fowler 2000] Fowler, Martin; Scott Kendall. 2000. *UML Distilled Second Edition, A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, Reading MA, USA.
- [Fujitsu 2002] Fujitsu. 2002. Homepage. <<http://www.fujitsu.com/>> (retrieved 19 February 2002).
- [Gamma 1995] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: elements of reusable object-oriented software*. 5th printing. Reading, MA, USA: Addison-Wesley Publishing Company, Inc.
- [Garrett 1998] Garrett, James H. Jr., Daniel P. Sieworiek, and Asim Smailagic. 1998. *Wearable Computer for Bridge Inspection*. Proceedings of the International Symposium on Wearable Computers (ISWC), Pittsburgh, PA, USA, October, 1998. pp. 160-161.
- [Garrett 2000] Garrett, James H. Jr. and Jirapon Sunkpho. 2000. *Issues in Delivering IT Systems to Field Users*. International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering, IKM 2000 Conference, June 22-24, Weimar, Germany.
- [Garrett 2002] Garrett, James H., Jr., Christian Bürgy, Jan Reinhardt, Jirapon Sunkpho. 2002. *An Overview of the Research in Mobile/Wearable Computer-Aided Engineering Systems in the Advanced Infrastructure Systems Laboratory at Carnegie Mellon University*. Bauen mit Computern 2002, Bonn, Germany. VDI Verlag GmbH, Duesseldorf.
- [GT Wearables] GT Wearables. 2002. Homepage. Georgia Institute of Technology Atlanta, GA, USA. <<http://wearables.gatech.edu/>> (retrieved 19 February 2002)
- [Gudgeirsson 2000] Guðgeirsson Garðar. 2000. *Requirements Engineering and XML*. Thesis, University of York, York, UK. <<http://www.rqml.org/>> (retrieved 28 October 2001)

References

- [Gulliksen 1997] Gulliksen, J., Lif, M., Lind, M., Nygren, E., & Sandblad, B. 1997. *Analysis of Information Utilisation*. International Journal of Human-Computer Interaction, 9 (3), 255-282.
- [Hammad 2002] Hammad, A., J. H. Garrett, H.A. Karimi. 2002. *Potential of Mobile Augmented Reality for Infrastructure Field Tasks*. 7th International Conference on 3D Web Technology: Tempe, AZ, USA.
- [Handspring 2002] Handspring. 2002. Homepage. <<http://www.handspring.com/>> (retrieved 19 February 2002).
- [Handykey 2002] Handykey Corporation. 2002. *Twiddler 2*. <<http://www.handykey.com/site/twiddler2.html>> (retrieved 10 January 2002)
- [Hewlett-Packard 2002] Hewlett-Packard. 2002. Homepage. <<http://www.hewlett-packard.com/>> (retrieved 19 February 2002).
- [Hurri 2000] Hurri, Jarmo. 2000. *Using decision tools in deciding system product requirements: literature review and a behaviourally motivated lightweight tool*. Helsinki University of Technology, Department of Computer Science and Engineering, Thesis. Helsinki, Finland.
- [IBM 2002] IBM. 2002. Homepage. <<http://www.ibm.com/>> (retrieved 19 February 2002).
- [ifcXML 2001] United Kingdom Chapter of the International Alliance for Interoperability (IAI). 2001. *IfcXML*. <http://cig.bre.co.uk/iai_uk/IFCXML.htm> (retrieved 19 December 2001).
- [Jacobson 1992] Jacobson Ivar. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading, MA, USA: Addison-Wesley.
- [Johnson 1992] Johnson, Peter. 1992. *Human Computer Interaction; Psychology, Task Analysis and Software Engineering*. McGraw-Hill Book Company, Berkshire, England.
- [Johnson 1997] Johnson, R. E. 1997. *Frameworks = (Components + Patterns)*. Communications of the ACM 40 (October): 39-42.
- [Kamm 1994] Kamm C. 1994. *User Interfaces for Voice Application*. In D.B. Roe and J. G. Wilpon (Eds), *Voice Communication Between Human and Machines*, pp. 422-42
- [Keller 1990] Keller, L., and Ho, J. 1990. *Decision-problem structuring*. In Concise Encyclopedia of Information Processing in Systems & Organizations, A. P. Sage, Ed. Pergamon Press. pp. 103-110.

References

- [Kieras 1997] Kieras, David E. 1997. *Task Analysis and the Design of Functionality*. The Computer Science and Engineering Handbook. Allen B. Tucker, Jr (editor-in-chief). Boca Raton, FL, USA. 1401-1423
- [Klinker 2001] Klinker Gudrun, Oliver Creighton, Allen H. Dutoit, Rafael Kobylinski, Christoph Vilsmeier, Bernd Brügge. 2001. *Augmented maintenance of powerplants: A prototyping case study of a mobile AR system*. IEEE and ACM International Symposium on Augmented Reality (ISAR'2001). New York, NY, USA, October 29-30, 2001.
- [Kruchten 2000] Kruchten, Philippe. 2000. *The Rational Unified Process – An Introduction*. Second Edition, 5th printing. Boston, MA, USA. Addison-Wesley.
- [L3 Systems 2002] L3 Systems. 2002. *WristPC keyboard*. <<http://www.l3sys.com/keybd/keybd.html>> (retrieved 10 January 2002)
- [Leffingwell 2000] Leffingwell, Dean and Don Widrig. 2000. *Managing Software Requirements – A Unified Approach*. Addison-Wesley.
- [Lif 1998] LIF, Magnus. 1998. *User Interface Modelling - adding usability to use cases*. Rep. No. 84, CMD, Uppsala, Sweden: Uppsala University.
- [m/w-CAE Systems 2002] Mobile and Wearable Computer-Aided Engineering Systems Lab, Carnegie Mellon University, Pittsburgh, PA, USA. 2002. <<http://www.ce.cmu.edu/~wearables>> (retrieved 20 January 2002)
- [Malaka 1999] Malaka, Rainer. 1999. *Deep Map: The Multilingual Tourist Guide*. Future Services For Networked Devices (FuSeNetD) – Workshop, Heidelberg, Germany, November 8-9, 1999.
- [Mann 1999] Mann Steve. 1999. *Humanistic Intelligence: WearComp as a new framework for Intelligent Signal Processing*. Proceedings of the IEEE, Vol. 86, No. 11, November 1998, Pages 2123-2151.
- [Mann 2001] Mann, Steve, Hal Niedzviecki. 2001. *Cyborg: Digital Destiny and Human Possibility in the Age of the Wearable Computer*. Randomhouse (Doubleday).
- [Mann 2001a] Mann, Steve. 2001. Homepage. <<http://www.eyetap.org/mann/>> (retrieved 12 December 2001)
- [Mann 2002] Mann, Steve. Personal e-mail. (retrieved 22 March 2002)
- [Markowitz 1996] Markowitz, Judith. 1996. *Using Speech Recognition*. Prentice Hall; Upper Saddle River, NJ, USA.

References

- [Meissner 2001] Meissner, Iris. 2001. *Development of a Concept for the Use of Mobile IT Devices for On-Site Data Collection Concerning Landfill Monitoring and the Prognosis of Landfill Reactions*. Thesis. Institute for Numerical Methods and Informatics in Civil Engineering, Technical University of Darmstadt, Germany.
- [Meissner 2002] Meissner, Iris and James H. Garrett, Jr. 2002. *Computer Aided Navigation in Unstructured Environments*. Working Paper.
- [MIT Wearables 2002] MIT Wearable Computing. 2002. Homepage. Massachusetts Institute of Technology, Cambridge, MA, USA.
<<http://www.media.mit.edu/wearables/>> (retrieved 19 February 2002)
- [MTO 2002] MTO, Micro Technology Office, DARPA. 2002. *Wearable Computer Systems with Head-Mounted Displays for Manufacturing, Maintenance, and Training Applications*.
<<http://www.darpa.mil/mto/smartmod/presentation/factsheets/boeingtrp.html>> (retrieved 13 February 2002).
- [MTO 2002a] MTO, Micro Technology Office, DARPA. 2002. *Wearable Maintenance Assitant*.
<<http://www.darpa.mil/mto/smartmod/presentation/factsheets/maintenance.html>> (retrieved 13 February 2002).
- [Myers 1998] Brad A. Myers. 1998. *A Brief History of Human Computer Interaction Technology*. ACM interactions. Vol. 5, no. 2, March, 1998. pp.44-54.
- [Najjar 1997] Najjar, L. J., J. C. Thompson, and J. J. Ockerman. 1997. *A wearable computer for quality assurance inspectors in a food processing plant*. In Digest of papers. First International Symposium on Wearable Computers pp. 163-164. Los Alamitos, CA, USA: IEEE Computer Society.
- [Nasbaum 1995] Nasbaum, H. C., Degroot, J., Lee, L. 1995. *Using speech Recognition Systems: Issues in Cognitive Engineering*. In Applied Speech Technology, CRC Press.
- [Niklfeld 2001] Niklfeld, G., R. Finan, M. Pucher. 2001. *Multimodal Interface Architecture for Mobile Data Services*. Proceedings of TCMC2001 Workshop on Wearable Computing, Graz, Austria.
- [Noah 2002] NOAH. 2002. *Notfall Organisations- und Arbeitshilfe*. <<http://www.uni-regensburg.de/Fakultaeten/Medizin/Uch/noah/>> (retrieved 20 February 2002)

References

- [Ockerman 1998] Ockerman, J., J., Thompson, J. C., and Najjar, L. J. 1998. *Wearable computers for manufacturing applications*. In Proceedings of Flexible Automation and Intelligent Manufacturing.
- [Oscar 1998] Oscar: Offshore Supply Crane Assistance Resource. 1998. *Conceptual Design Presentation*. June 26, 1998 at Chevron Corporation Houma, LA, USA.
- [Palm 2002] Palm. 2002. Homepage. <<http://www.palm.com>> (retrieved 19 February 2002).
- [PC-EPhone 2002] PC-EPhone. 2002. Homepage. <<http://www.pcephone.com/>> (retrieved 19 February 2002).
- [Piekarski 2001] Piekarski, Wayne and Bruce H. Thomas. 2001. *Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer*. Proceedings of The Fifth International Symposium on Wearable Computers, Zürich, Switzerland, 8-9 October 2001.
- [Rational 2001] Rational Software. 2001 *Rational Unified Process for Systems Engineering RUP SE1.0 A Rational Software White Paper TP 165, 8/01*. Cupertino, CA, USA.
- [Ravey 2002] Ravey, Donald. 2002. *Microsoft Access 2000 Relational Database Management System*. <<http://www.ravey.net/Skyline/msaccess.html>> (retrieved 17 February 2002).
- [Regnell 1999] Regnell, Björn. 1999. *Requirements Engineering with Use Cases – a Basis for Software Development*. Thesis, Lund University, Department of Communication Systems, Lund Institute of Technology.
- [Reinhardt 2000] Reinhardt, Jan, James H. Garrett Jr., Raimar J. Scherer. 2000. *The preliminary design of a wearable computer for supporting Construction Progress Monitoring*. International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering (IKM 2000). Weimar, Germany.
- [Roeckelein 2002] Röckelein, Wolfgang. 2002. Personal e-mail. (retrieved 27 February 2002)
- [Rosenfeld 2000] Rosenfeld, R., Zhu, X., Toth, A., Shriver, S., Lenzo, K. and Black, A. 2000. *Towards a Universal Speech Interface*. ICSLP2000, Beijing, China.

References

- [Rudnicky 1996] Rudnicky, Alexander I. *Speech Interface Guidelines*. <<http://www.speech.cs.cmu.edu/rspeech-1/air/papers/SpInGuidelines/SpInGuidelines.html>> (retrieved 20 March 2001)
- [Rumbaugh 1991] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. 1991. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ, USA: Prentice Hall.
- [Sackin 1999] Sackin, Douglas, Jirapon Sunkpho, Jason Small, James H. Garrett, Jr., Daniel P. Siewiorek, Asim Smailagic. 1999. *Design of Computer Support for Field Operations*. Proceedings of International Association of Bridge and Structural Engineers Conference, Rio de Janeiro Brazil, August, 1999
- [Sage 1991] Andrew P. Sage. 1991. *Decision Support Systems Engineering*. John Wiley & Sons, Inc., New York, USA.
- [Schaechinger 2000] Schächinger, U., R. Kretschmer, W. Röckelein, C. Neumann, M. Maghsudi, M. Nerlich. 2000 NOAH – A Mobile Emergency Care System. *NOAH - A Mobile Emergency Care System*. In: European Journal of Medical Research, 1/2000, p. 13-18.
- [Shirogane 1998] Shirogane, Junko; Yoshiaki Fukazawa. 1998. *Method of User-Customizable GUI Generation and Its Evaluation*. 5th Asian Pacific Software Engineering Conference, Taipei, Taiwan.
- [Shneiderman 1997] Shneiderman, Ben. 1997. *Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces*. <<http://citeseer.nj.nec.com/shneiderman97direct.html>> (retrieved 12 December 2001).
- [Siegel 1997] Siegel, J. and M. Bauer. 1997. *A field usability evaluation of a wearable system*. In Proceedings of The First International Symposium on Wearable Computers, Cambridge, MA, USA, October, 1997, pp. 18-22.
- [Siewiorek 1998] Siewiorek, D., A. Smailagic, L. Bass, J. Siegel, R. Martin, B. Bennington. 1998. *Adtranz: A Mobile Computing System for Maintenance and Collaboration*. In Proceedings of the Second International Symposium on Wearable Computers (October).
- [Smailagic 1998] Smailagic, Asim. 1998. *An Evaluation of Audio-Centric CMU Wearable Computers*. ACM Journal on Special Topics in Mobile Networking and Applications, Vol. 6, No. 4, 1998, pp. 65-76.

References

- [Smailagic 1999] Smailagic, Asim, Dan Siewiorek. 1999. *User-Centered Interdisciplinary Design of Wearable Computers*. ACM Mobile Computing and Communications Review, Vol. 3, No. 3, pp 43-52.
- [Smailagic 2000] Smailagic, A., Siewiorek, D.P., Reilly, D. 2000. *Power and Performance Analysis of CMU Wearable Computers*. Proceedings IEEE Symposium on Wearable Computers, Atlanta, GA, USA. October 2000.
- [Smailagic 2001] Smailagic, A., Siewiorek, D. P., Ettus, M. 2001 *System Design of Low - Energy Wearable Computers with Wireless Networking.*, Proceedings IEEE Symposium on VLSI, Orlando, FL, USA. April 2001.
- [SMALTO 1999] SMALTO. 1999. *Welcome to SMALTO: the Speech Modality Auxiliary Tool*. <<http://disc.nis.sdu.dk/smalto/>> (retrieved 11 November 2000).
- [Stamer 1998] Stamer, T., B. Schiele, A. Pentland. 1998 *Visual Contextual Awareness in Wearable Computing*. Proceedings 2nd International Symposium Wearable Computers (ISWC 98), IEEE CS Press, Los Alamitos, CA, USA, pp. 50–57.
- [Stars 2001] Stars: Sticky Technology for Augmented Reality Systems. 2001. Course homepage, joined course between Carnegie Mellon University, Pittsburgh, PA, USA and Technical University, Munich, Germany. <<http://stars.globalse.org/STARS2001/>> (retrieved 15 January 2001)
- [Stumpf 1998] Stumpf, Annette, Liang Y. Liu, Chul-Soo Kim, and Sangyoon Chin. 1998. *Delivery and Test of the Digital Hardhat System at U.S. Army Corps of Engineers Fort Worth District Office*. US Army Corps of Engineers Construction Engineering Research Laboratories. USACERL ADP Report 99/16, December 1998.
- [Sun 1998] Sun Microsystems. 1998. *Java™ Speech API Programmer's Guide Version 1.0*. October 26, 1998, Palo Alto, CA, USA.
- [Sunkpho 1998] Sunkpho, Jirapon, James H. Garrett, Jr., Asim Smailagic, Dan Siewiorek. 1998. *MIA: A Wearable Computer for Bridge Inspectors*. Proceedings, IEEE, The Second International Symposium on Wearable Computers, Pittsburgh, PA, USA 1998.
- [Sunkpho 2000] Sunkpho, Jirapon, James H. Garrett, Jr., and Asim Smailagic. 2000. *Opportunities to use Speech Recognition for Bridge Inspection*. In the Proceedings of the 2000 ASCE Construction Congress, Orlando, FL, USA, Feb 20-22. pp. 184-193.

References

- [Sunkpho 2001] Sunkpho, Jirapon. 2001. *A Framework for Developing Field Inspection Support Systems*. Ph.D. thesis. Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA, USA.
- [Tangis 2002] Tangis. 2002. Tangis In-Motion Development Suite 2.0. <<http://www.tangis.com/solutions/devsuite.htm>> (retrieved 20 February 2002)
- [Thompson 2000] Thompson, Cynthia, A. and Mehmet H. Göker. 2000. *Learning to Suggest: The Adaptive Place Advisor*. Proceedings of the 2000 AAAI Spring Symposium on Adaptive User Interfaces. Menlo Park, California, USA.
- [Thorp 1998] Thorp, Edward O. 1998. *The Invention of the First Wearable Computer*. Proceedings of the Second International Symposium on Wearable Computers (ISWC), Pittsburgh, PA, USA, October, 1998. pp. 4-8.
- [TUV 2002] TÜV Rheinland Japan. 2002. Ingress Protection (IP) Codes. <http://www.jpn.tuv.com/services/machinery_components/lab_s_ip_tests.shtml> (retrieved 23 February 2002)
- [Van Dam 1997] Van Dam, Andries. 1997. *Post-WIMP User Interfaces*. Communications of the ACM, February 1997, Vol. 40, No. 2, pp.63-67.
- [Van Harmelen 2001] Van Harmelen, Mark; editor. 2001. *Object Modeling and User Interface Design*. Boston, MA, USA: Addison-Wesley.
- [ViA 2002] ViA. 2002. Homepage. <<http://www.via-pc.com/>> (retrieved 19 February 2002).
- [Vo 1998] Vo, Minh Tue. 1998. *A Framework and Toolkit for the Construction of Multimodal Learning Interfaces*. Ph.D. thesis. School of Computer Science Computer Science Department Carnegie Mellon University Pittsburgh, PA, USA.
- [Vocollect 2001] Vocollect. 2001. *Voice power solution for business*. <<http://www.vocollect.com>> (retrieved November 4, 2001).
- [voiceXML 2000] VoiceXML Forum. 2000. <<http://www.voicexml.org/>> (retrieved 19 December 2001).
- [W3C 1997] World Wide Web Consortium (W3C). 1997. *Extensible Markup Language (XML)*. <<http://www.w3.org/XML/>> (retrieved 21 October 2001)
- [Walkabout 2002] Waikabout Computers. 2002. Homepage. <<http://www.walkabout-comp.com/>> (retrieved 19 February 2002).

References

- [Warmer 1998] Warmer, Jos B. and Anneke G. Kleppe. 1998. *The Object Constraint Language – Precise Modeling with UML*. Teading, MA, USA: Addison Wesley Longman, Inc.
- [WearableGroup 2002] WearableGroup. 2002. Homepage. Carnegie Mellon University, Pittsburgh, PA, USA. <<http://www.wearablegroup.org>> (retrieved 19 February 2002)
- [Wearix 2002] Wearix Software GmbH. 2002. Wearix Application Broker. <<http://www.wearix.com/uk/produkte/pr.html>> (retrieved 20 February 2002)
- [Wearlab 2002] Wearlab, TZI, Bremen, Germany. 2002. *Technologische und anwendungsorientierte Potenziale mobiler, tragbarer Computersysteme*. Research study. <<http://wearlab.informatik.uni-bremen.de/DOCS/studie/index.html>> (retrieved 13 February 2002).
- [Williamson 2001] Williamson, Heather. 2001, XML : the complete reference. Berkeley, CA, USA, London, U.K.: Osborne/McGraw-Hill.
- [Winspect 2002] Winspect. 2002. *Wartung, Instandhaltung und Inspektion großtechnischer Anlagen durch Einsatz mobiler Computertechnik*. <<http://www.tzi.de/wearable/DOCS/winspect.html>> (retrieved 14 February 2002)
- [XUL 2000] The XML Cover Pages. 2000. *Extensible User Interface Language (XUL)*. <<http://www.oasis-open.org/cover/xul.html>> (retrieved 19 December 2001).
- [Xybernaut 2001] Xybernaut Corporation. 2002. *Evolving Computing Technology with Real-World Product Innovations- Solutions that Provide a Competitive Edge and a Measurable Return on Investment*. <http://www.xybernaut.com/newxybernaut/learn_more/learn_2.htm> (retrieved 27 December 2001)
- [Xybernaut 2002] Xybernaut. 2002. Homepage. <<http://www.xybernaut.com>> (retrieved 19 February 2002).