

Otto-von-Guericke-Universität Magdeburg



Fakultät für Informatik  
Institut für Verteilte Systeme  
Arbeitsgruppe Softwaretechnik

# Diplomarbeit

**Konzeption und Implementation von palmbasierten  
Werkzeugen zur Unterstützung des  
Softwareentwicklungsprozesses**

Verfasser:

Daniel Reitz

30. Januar 2001

Betreuer:

Prof. Dr.-Ing. habil. Reiner R. Dumke

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik

Postfach 4120, D-39106 Magdeburg

**Reitz, Daniel:**

*Konzeption und Implementation von palm-  
basierten Werkzeugen zur Unterstützung des  
Softwareentwicklungsprozesses*

Diplomarbeit, Otto-von-Guericke-Universität  
Magdeburg, 2001.

## Zusammenfassung

Zielstellung der vorliegenden Arbeit ist es, Möglichkeiten für die Nutzung des Palm-Handheldcomputers, zum Einsatz auf dem Gebiet des " *Computer Aided Software Engineering (CASE<sup>1</sup>)*", zu erarbeiten. Hintergrund der Zielsetzung sind die Vorteile des Palms; wie seine ständige Verfügbarkeit, seine kommunikativen Fähigkeiten und seine große Verbreitung. Die Diplomarbeit kann im Bereich der Softwaretechnik (eng. Software Engineering) eingeordnet werden.

Der Aufgabenschwerpunkt liegt in der Entwicklung von konkrete Ideen für palmbasierte Werkzeuge zur Unterstützung des Softwareentwicklungsprozesses und im Entwurf von einigen geeignete Beispielen. Dazu werden die technologischen Grundlagen des Palms analysiert und der Einsatz von CASE-Tools im Lebenszyklus von Software-Projekten untersucht.

Das praktische Ergebnis der Diplomarbeit ist die Spezifikation, der Entwurf und die Implementation einer Palm-Applikation, zur Unterstützung der Aufwandschätzung von Software-Projekten, mittels der Function-Point Zählmethode.

Für den Entwurf der Werkzeuge kommt UML<sup>2</sup> (Unified Modeling Language) als Modellierungssprache zum Einsatz.

---

<sup>1</sup>siehe [Mar94a, Seite 75]

<sup>2</sup>siehe [SvG00]



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Einführung und Motivation . . . . .	1
1.2	Gliederung der Arbeit . . . . .	3
<b>2</b>	<b>Technologische Grundlagen des Palm Handheldcomputers</b>	<b>5</b>
2.1	Hardware-Merkmale . . . . .	5
2.2	Standard Software . . . . .	6
2.3	Kommunikation . . . . .	8
2.4	Anwendungsgebiete . . . . .	9
2.5	Die Palm Programmierung . . . . .	11
2.6	Zusammenfassung . . . . .	14
<b>3</b>	<b>CASE-Tools im Software-Lebenszyklus</b>	<b>17</b>
3.1	Die Software-Entwicklung, -Anwendung und -Wartung . . . . .	17
3.1.1	Die Problemdefinition . . . . .	17
3.1.2	Die Anforderungsanalyse . . . . .	18
3.1.3	Die Spezifikation . . . . .	20
3.1.4	Der Entwurf . . . . .	21
3.1.5	Die Implementation . . . . .	22
3.1.6	Die Erprobung . . . . .	24
3.1.7	Die Wartung . . . . .	24
3.1.8	Die Anwendung . . . . .	25
3.2	Projekt begleitende CASE-Tools . . . . .	26
3.2.1	Projektmanagement-Programme . . . . .	26
3.2.2	Programme zur Qualitätssicherung . . . . .	29
3.3	CASE-Umgebung . . . . .	31
3.4	Zusammenfassung . . . . .	32

<b>4</b>	<b>CASE-Tools für den Palm</b>	<b>37</b>
4.1	Eine metrikbasierten Erfahrungs-Datenbank für den Palm (P3)	37
4.1.1	Szenariobeschreibung . . . . .	39
4.1.2	Die Datenbank der Palm-Applikation (MetrikDB) . . .	42
4.1.3	Die Zustände der Metrik-Datenbank . . . . .	44
4.1.4	Die Komponenten und Klassen der Metrik-Datenbank	45
4.1.5	Fazit und Ausblick . . . . .	47
4.2	Ein palmbasiertes Formular zur Erstellung und Anzeige von Fehler-Protokollen (P4) . . . . .	48
4.2.1	Szenariobeschreibung . . . . .	50
4.2.2	Die Komponenten der Palmapplikation . . . . .	52
4.2.3	Die Klassen der Fehler-Protokoll Desktop-Applikation	53
4.2.4	Kommunikation . . . . .	53
4.2.5	Fazit und Ausblick . . . . .	54
4.3	Ein palmbasiertes CASE-Tool zur Unterstützung der ISO-9000 Zertifizierung (P7) . . . . .	55
4.3.1	Szenario Beschreibung . . . . .	55
4.3.2	Die Zustände des ISO-9000 Tools . . . . .	58
4.3.3	Die Komponenten des ISO-9000 Tools . . . . .	59
4.3.4	Fazit und Ausblick . . . . .	60
4.4	Ein palmbasiertes CASE-Tool zur Unterstützung der CMM-Methode (P8) . . . . .	60
4.4.1	Szenariobeschreibung . . . . .	62
4.4.2	Die Zustände und Komponenten des CMM-Tools . . .	64
4.4.3	Fazit und Ausblick . . . . .	64
<b>5</b>	<b>Spezifikation, Entwurf und Implementation eines palmbasier-ten Function-Point Zähltools</b>	<b>65</b>
5.1	Computergestützte Eingabe der Daten . . . . .	66
5.2	Die Anforderungen an das Function-Point Zähltool . . . . .	67

5.3	Die Datenbank des Function–Point Zähltools . . . . .	70
5.4	Szenariobeschreibung . . . . .	75
5.5	Die Zustände des Function–Point Zähltools . . . . .	83
5.6	Die Komponenten von FPC . . . . .	86
5.7	Die Klassen von FPCAnalyse . . . . .	88
5.8	Die Verteilung des Function–Point Zähltools . . . . .	90
5.9	Analyse des Quellcodes mit PC–Metrik . . . . .	90
5.10	Fazit und Ausblick . . . . .	91
<b>6</b>	<b>Zusammenfassung</b>	<b>93</b>
6.1	Ergebnisse . . . . .	93
6.2	Ausblick . . . . .	93
<b>A</b>	<b>Endnutzerdokumentation des Function–Point Zähltools</b>	<b>95</b>
A.1	Einleitung . . . . .	95
A.1.1	Funktionsweise . . . . .	95
A.1.2	Die Elemente der verschiedenen Datensätze . . . . .	97
A.2	Die Bedienung der Palm Applikation . . . . .	98
A.2.1	Erstellung eines neuen Projektes . . . . .	98
A.2.2	Eingabe der Daten . . . . .	99
A.2.3	Modifizierung von Datensätzen . . . . .	102
A.2.4	Auswertung . . . . .	103
A.3	Die Bedienung der Windows Applikation . . . . .	104
<b>B</b>	<b>Beispiel Function–Point Zählung</b>	<b>109</b>
B.1	Resultate . . . . .	110
B.2	Fazit . . . . .	117
<b>C</b>	<b>Ausschnitte aus dem FPC–Sourcecode</b>	<b>119</b>
	<b>Literaturverzeichnis</b>	<b>125</b>

<b>Selbständigkeitserklärung</b>	<b>128</b>
<b>Thesen</b>	<b>129</b>

## Abbildungsverzeichnis

1	Der Palm IIIx . . . . .	6
2	Die Softwarepackages . . . . .	7
3	Kommunikation zwischen Palm und PC . . . . .	8
4	Die Anwendungsgebiete . . . . .	10
5	Die Source-Files für das GNU-Pilot-SDK . . . . .	12
6	Das GNU-Pilot-SDK . . . . .	14
7	Netzplanarten nach [Bal98] . . . . .	27
8	Screenshots von <i>Project Plan</i> (l.) und <i>Project Punch</i> (r.) . . . . .	28
9	NIST/ECMA Referenzmodell einer CASE-Umgebung, siehe [Pfl99] . . . . .	31
10	Anwendungsfälle der Metrik-Datenbank . . . . .	39
11	Einstellungen zur Eingabe der Messdaten . . . . .	40
12	ERM-Modell der MetrikDB . . . . .	43
13	Die Zustände der Palm-Applikation (MetrikDB) . . . . .	45
14	Die Komponenten der Palm-Applikation . . . . .	46
15	Anwendungsfälle des Fehler-Protokoll Editors . . . . .	50
16	Bildschirme zur Erstellung eines neuen Protokolls . . . . .	51
17	Die Komponenten der Palmapplikation . . . . .	52
18	Die Klassen der Fehler-Protokoll Desktop-Applikation . . . . .	53
19	Die Kommunikation zwischen zwei Palm-Computern . . . . .	54
20	Anwendungsfälle des ISO-9000 Tools . . . . .	56
21	Form zur Auswahl der Kategorie . . . . .	57
22	Form Beantwortung der Fragen . . . . .	58
23	Die Zustände des ISO-9000 Tools (aus Nutzersicht) . . . . .	59
24	Das Komponentendiagramm des ISO-9000 Tools . . . . .	60
25	Startbildschirm der CMM-Applikation . . . . .	62
26	Auswahlbildschirm zu den fünf Stufen (Level) der CMM-Applikation . . . . .	63

27	Die Funktionstypen der FFP-Methode nach IFPUG . . . . .	66
28	Die Komplexitätsmatrix einer Funktion vom Typ "EIF" . . . .	67
29	Eine Wertetabelle für die FPA Methode . . . . .	68
30	Das ER-Modell der FPC-Datenbank . . . . .	70
31	Die Reihenfolge der Records in der FPC-Datenbank . . . . .	71
32	Die Verknüpfungen der Records der FPC-Datenbank . . . . .	74
33	Das Anwendungsfalldiagramm von FPC . . . . .	75
34	Die Bildschirme zur Erstellung eines neuen Projektes . . . . .	76
35	Das Sequenzdiagramm zur Erstellung neuer Datensätze . . . .	77
36	Beispiel für eine Komplexitätsmatrix und eine Wertetabelle in FPC . . . . .	78
37	Beispielbildschirme zum "Browsen" durch die FPC-Datenbank	79
38	Bildschirm zur Auswahl des zu analysierenden Datensatzes . .	80
39	Bildschirme zur Auswertung der Datensätze . . . . .	80
40	Windowsapplikation zur Auswertung und Berichtgenerierung .	82
41	Zustandsdiagramm zur Erstellung eines neuen Projektes . . . .	83
42	Zustandsdiagramm zur Eingabe der Funktionstypen . . . . .	84
43	Zustandsdiagramm zur Navigation durch die FPC-Datenbank	85
44	Zustandsdiagramm zur Auswertung von Datensätzen . . . . .	86
45	Zustandsdiagramm von FPCAnalyze . . . . .	86
46	Die Komponenten des Function-Point Zähltools . . . . .	87
47	Das Klassendiagramm von FPCAnalyze . . . . .	89
48	Das Verteilungsdiagramm des Function-Point Zähltools . . . .	90
49	Bildschirm zur Erstellung eines neuen Prozessdatensatzes . . .	99
50	Bildschirme zur detaillierten Eingabe der Kontroll Funktions- typen . . . . .	101

## Abkürzungsverzeichnis

<b>ASCII</b>	American Standard Code for Information Interchange
<b>CASE</b>	Computer Aided Software Engineering
<b>CMM</b>	Capability Maturity Model
<b>CoCoMo</b>	Constructive Cost Model
<b>COSMIC</b>	Common Software Measurement International Consortium
<b>DB</b>	Datenbank
<b>DET</b>	Data Element Type
<b>ECE</b>	External Control Entry
<b>ECX</b>	External Control Exit
<b>EI</b>	External Input
<b>EIF</b>	External Interface File
<b>EO</b>	External Output
<b>EQ</b>	External Inquire
<b>ERM</b>	Entity Relationship Model
<b>FFP</b>	Full-Function-Point Method
<b>FP</b>	Function-Point
<b>FPA</b>	Function-Point Analyse
<b>FPC</b>	Function-Point Counter
<b>GPL</b>	GNU Public License
<b>GPS</b>	Global Positioning System
<b>HTML</b>	Hypertext Markup Language
<b>ICR</b>	Internal Control Read
<b>ICW</b>	Internal Control Write
<b>IFPUG</b>	International Function-Point UserGroup
<b>ILF</b>	Internal Logical File
<b>IP</b>	Internet Protocol
<b>ISO</b>	International Standard Organization
<b>KB</b>	Kilo Byte
<b>LOC</b>	Lines Of Code
<b>MB</b>	Mega Byte
<b>NIST</b>	National Institute of Standards and Technology
<b>OS</b>	Operating System
<b>PC</b>	Personal Computer
<b>PIM</b>	Personal Information Management
<b>POSE</b>	PalmOS Emulator
<b>QA</b>	Question and Answer

<b>RAM</b>	Random Access Memory
<b>RCG</b>	Read-only Control Group
<b>RET</b>	Record Element Type
<b>ROM</b>	Read-only Memory
<b>RTF</b>	Rich Text Format
<b>SDK</b>	Software Development Kit
<b>TCP</b>	Transport Control Protocol
<b>UCG</b>	Updated Control Group
<b>UFP</b>	Unadjusted Function-Points
<b>UI</b>	User Interface
<b>UML</b>	Unified Modeling Language
<b>VAF</b>	Value Adjustment Factor
<b>WAP</b>	Wireless Access Protocol
<b>WWW</b>	World Wide Web

# 1 Einleitung

## 1.1 Einführung und Motivation

Das Gebiet der Softwaretechnik (eng. Software Engineering) beschäftigt sich im weitesten Sinne mit der Entwicklung, Anwendung und Wartung von Software-Produkten<sup>3</sup>. Dazu gehören Methoden, Werkzeuge, Maßsysteme, Personen, Standards und Erfahrungen.

Definiert wird die Softwaretechnik in [Mar94b, Seite 1177] wie folgt:

„**Software-Engineering:** *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software.*“

Der Software-Entwicklungsprozess, die Wartung und Anwendung bilden den Lebenszyklus eines Software-Produktes, welcher in folgende Phasen unterteilt werden kann:

- Software-Entwicklung
  - Problemdefinition
  - Anforderungsanalyse
  - Spezifikation
  - Entwurf
  - Implementation
  - Erprobung
  - Auslieferung
- Software-Anwendung
  - Einführung
  - Nutzung
  - Umstellung
  - Ablösung

---

<sup>3</sup>auch Applikationen oder Programme

- Software–Wartung
  - Übernahme
  - Änderung
  - Versionserstellung
  - Verteilung

Eine ausführliche Beschreibung der einzelnen Lebenszyklus–Phasen befindet sich in [Dum00, ab Seite 17].

Bei den Werkzeugen, die zur Unterstützung der Entwicklung, Anwendung und Wartung von Software–Systemen eingesetzt werden, handelt es sich um sogenannte CASE–Tools (*CASE = Computer Aided Software–Engineering*). [Tha97, Seite 509] definiert den Begriff wie folgt:

*„CASE tool—An automated software engineering development tool that can assist software engineering in analyzing, designing, coding, testing and documenting a software system and managing a software project. John Manley, University of Pittsburgh, is apparently the first person to use the acronym CASE for computer–aided software engineering.“*

Beispiele für solche Tools sind Modellierungs–Programme, Integrierte Entwicklungs–Umgebungen, Messtools, Projektmanagement–Programme, Compiler, Debugger usw.

Die Arbeitsgruppe Softwaretechnik, des Institutes für Informatik, an der Universität Magdeburg, unter der Leitung von Prof. Reiner R. Dumke, konzentriert sich schwerpunktmäßig auf den Bereich der Software–Messung<sup>4</sup>. Hier entstand die Idee für die Entwicklung eines CASE–Tools zur Unterstützung der Aufwandschätzung von Software–Projekten mittels der Function–Point–Methode, für den Palm–Handheldcomputer, welches auch 1999 im Rahmen meines Laborpraktikums als Prototyp realisiert wurde. Die Wahl fiel dabei auf den Mobil–Computer der Firma Palm Inc.<sup>5</sup> (ehemals zu 3Com gehörend), aufgrund seiner enormen Verbreitung (Handheldcomputer mit dem PalmOS–Betriebssystem haben z.Z. einen Marktanteil von über siebenzig Prozent), seiner Bedienerfreundlichkeit und seiner einfachen Erweiterbarkeit mit eigener oder fremder Soft– und Hardware.

---

<sup>4</sup><http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe>

<sup>5</sup><http://www.palm.com>

Nach der erfolgreichen Fertigstellung des Prototypen, stellte sich die Frage, ob für den Palm noch weitere Einsatzmöglichkeiten auf dem Gebiet der computergestützten Software-Entwicklung existieren. Damit war Grundlage für diese Diplomarbeit geschaffen.

Des weiteren soll im Rahmen dieser Arbeit, der Prototyp des Function-Point Zähltools zur Produktreife weiterentwickelt werden, was u.a. im Interesse der “Software Measurement Technology (SMT) GmbH”, unter der Leitung von Herbert H. W. Telieps<sup>6</sup>, an einer solchen Lösung begründet lag. Profitieren konnte ich dabei durch das Engagement der Softwaretechnik Arbeitsgruppe in der Verbreitung und Weiterentwicklung der Function-Point Methode und einer daraus resultierenden HiWi-Tätigkeit, die in Zusammenarbeit mit der “Siemens AG Automotiv” das Ziel hatte, Produktspezifikationen nach der Function-Point Methode zu analysieren (siehe [DFS99]).

## 1.2 Gliederung der Arbeit

Das **erste Kapitel** gibt einen Einblick in den Bereich des “*Computer Aided Software Engineering*” und versucht für den Einsatz des Palm-Handheldcomputers auf diesem Gebiet zu motivieren.

Im **zweiten Kapitel** wird der Palm-Handheldcomputer mit seiner Hard- und Softwareausstattung vorgestellt und seine Anwendungsgebiete beschrieben.

**Kapitel 3** untersucht jede einzelne Lebenszyklusphase von Software-Produkten auf Aktionen und Methoden sowie die Verwendung von CASE-Tools und diskutiert dabei den möglichen Einsatz des Palm-Handheldcomputers.

Gegenstand von **Kapitel 4** ist der Entwurf einiger Beispiel-Lösungen für palmbasierte CASE-Tools aus dem dritten Kapitel.

In **Kapitel 5** wird ein im Rahmen der Diplomarbeit entwickeltes und implementiertes CASE-Tool, zur Unterstützung des Function-Point Zählprozesses, spezifiziert und entworfen.

**Kapitel 6** enthält eine Zusammenfassung der Ergebnisse der vorliegenden Arbeit und gibt einen Ausblick auf mögliche Weiterentwicklungen dieses Projektes.

Der **Anhang** enthält die Endnutzerdokumentation des Function-Point-Zähltools, die Ergebnisse der Untersuchung des Programmentwurfs mit der Function-Point Methode und einige Ausschnitte aus dem Sourcecode.

---

<sup>6</sup><http://www.ubt.de>



## 2 Technologische Grundlagen des Palm Handheldcomputers

Ein Handheldcomputer, wie der Palm, hat systembedingt spezifische Vor- und Nachteile gegenüber herkömmlichen Desktop- und Mobilcomputern. Es ist Aufgabe dieses Kapitels die technischen Grundlagen vorzustellen und zu analysieren, um eine Grundlage zu schaffen die es erlaubt, die Einsatzmöglichkeiten des Palms beim computergestütztem Software Engineering zu eruieren.

Der Palm ist am einfachsten als elektronischer, tastaturloser Organizer zu beschreiben. Seine Softwareausstattung umfasst Programme um Adressen und Notizen zu speichern, er kann Termine sowie Aufgabenlisten verwalten und er besitzt die Funktionalität eines einfachen Taschenrechners.

Was ihn aber von einfachen Organizern unterscheidet, ist neben dem Preis, die Möglichkeit ihn mit weiteren Programmen zu versorgen. Diese kann man entweder selber entwickeln oder man besorgt sie sich von Drittanbietern.

Ein weiteres wichtiges Merkmal des Palms ist seine Fähigkeit mit anderen Computern zu kommunizieren, entweder über Infrarot, um vorwiegend mit anderen Palms Daten auszutauschen oder seriell, um z.B. auf einem Desktop Computer Daten zu sichern oder um Programme zu laden. Da das Betriebssystem des Palms (PalmOS) einen kompletten TCP/IP Stack beinhaltet, stehen dem Nutzer, mit einem zusätzlichen Modem und entsprechender Software, sogar die Möglichkeiten des Internets zur Verfügung.

Damit werden neben der reinen Organizerfunktionalität des Palm noch weitere Anwendungsgebiete ersichtlich (siehe Abschnitt 2.4).

### 2.1 Hardware-Merkmale

Die verschiedenen Modelle der Palm-Familie sind mit folgender Hardware ausgestattet:

- **Prozessor:** Motorola 68k (bis 32MHz),
- **RAM:** 128k bis 8MB,
- **ROM:** aktuell 1MB,
- **Display:** 160x160 Bildpunkte mit bis zu 16 Graustufen oder 256 Farben (berührungssensitiv),



Abbildung 1: Der Palm IIIx

- **serieller Port:** zum Datenaustausch, über serielles Kabel oder Modem mit einem PC oder Internet,
- **Infrarot Port:** zum Datenaustausch zwischen zwei Palm-Computern (ab Palm III).

Die Abbildung 1 zeigt einen Palm IIIx. Der Rahmen enthält Knöpfe zum An- und Ausschalten des Palms, zum Aufruf der Standard-Applikationen und zum Hoch- und Runterscrollen. Im Eingabebereich können Buchstaben, Zahlen und Zeichen durch den Nutzer eingegeben werden, die das Betriebssystem interpretieren soll. Außerdem befinden sich dort noch vier Buttons, die mit speziellen Funktionen vorbelegt sind (siehe auch [Pog98, Car98]). Den meisten Knöpfen und Buttons lässt sich aber auch eine andere Funktion zuordnen.

## 2.2 Standard Software

Die Software des Palm Handheldcomputers besteht standardmäßig aus dem Betriebssystem, ein paar Systemprogrammen und den Standardapplikationen. Diese Programme liegen alle komplett im ROM des Palm. Alle Programme, die sich der Nutzer zusätzlich besorgt, werden im RAM abgelegt und verbleiben dort auch solange, bis sie entweder wieder gelöscht werden

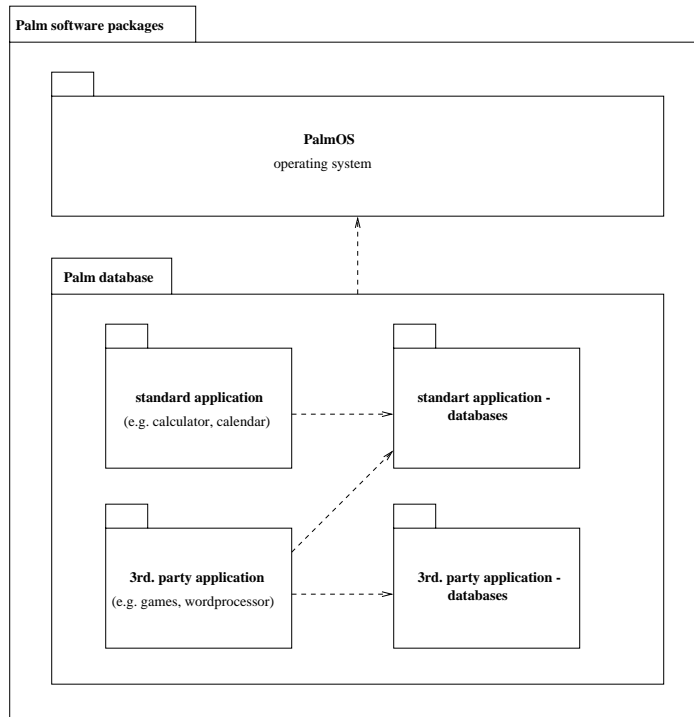


Abbildung 2: Die Softwarepackages

oder die Batterien bzw. der Akku nicht mehr genügend Strom liefern. Der Palm wird mit folgenden Applikationen ausgeliefert:

- Terminkalender,
- Adressdatenbank,
- Aufgabenliste,
- Notizbuch,
- Taschenrechner,
- Mailreader und
- Kostenverwaltung.

Weiterhin stehen dem Nutzer noch Programme zum Einstellen von Systemkonfigurationen, zur Kommunikation und zum Erlernen der Zeichen des

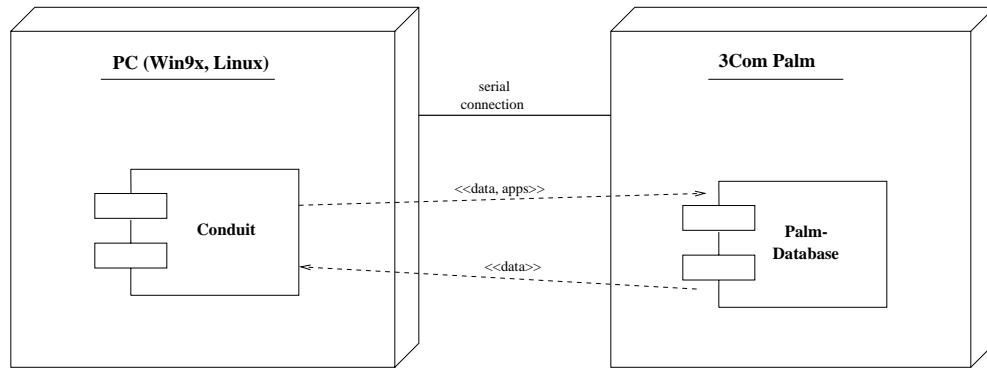


Abbildung 3: Kommunikation zwischen Palm und PC

Palm-Alphabetes zur Verfügung.

Alle Programme, ob Standardapplikationen oder Programme von Drittanbietern, werden im Palm in einer Datenbank verwaltet, dabei sind ausführbarer Code und Daten voneinander getrennt (siehe Abbildung 2).

Zur Synchronisation des Datenbestandes der Standardapplikationen mit einem PC (siehe Abschnitt 2.3) existieren sogenannte PIM-Programme (PIM steht für Personal Information Management). Sie ermöglichen eine verteilte Eingabe der Daten entweder auf dem Palm oder dem Desktop-Computer und sorgen beim Hotsync-Vorgang (der Palm wird über seine Dockingstation mit dem PC verbunden) für einen Abgleich der Datenbanken. Die von Palm Computing mitgelieferte PIM-Software heißt *Palm Desktop* und bedient alle Standardapplikationen (siehe oben). Daneben gibt es eine grosse Zahl weiterer Anwendungen, die PIM-Programme für den Palm beinhalten. Dazu zählen unter anderem: *MS Outlook Express*<sup>7</sup>, *Staroffice*<sup>8</sup> und der *KOrganizer* des KDE-Projektes<sup>9</sup>.

## 2.3 Kommunikation

Der Palm kann auf zwei verschiedenen Wegen mit anderen Computern kommunizieren:

1. Seriell, über ein Kabel oder Modem mit einem PC. Dazu existiert spezielle Kommunikationssoftware, für die Standardapplikationen PIM-

<sup>7</sup><http://www.microsoft.com>

<sup>8</sup><http://www.sun.com>

<sup>9</sup><http://www.kde.org>

Programme und für Applikationen von Drittanbietern sogenannte Conduits, um den Datenaustausch zu ermöglichen (siehe Abbildung 3).

2. Über die Infrarotschnittstelle mit anderen Palm-Computern. Allerdings ist dies standardmäßig erst ab dem Palm III möglich, alle früheren Versionen benötigen ein Hardware-Upgrade.  
Mit spezieller Software kann der Palm über Infrarot auch mit einem PC Daten austauschen, vorausgesetzt der PC besitzt eine Infrarot-Schnittstelle (aktuelle Notebooks sind häufig damit ausgestattet).

Die Kommunikation ist eins der wichtigsten Features des Palms. Sie dient zur Sicherung von wichtigen Daten (Adressen, Kostenplänen, Merktzettel usw.) und zur Speicherung von neuen Programmen und unterwegs benötigten Informationen.

Wenn an den Palm ein analoges oder digitales Modem angeschlossen wird, kann der Palm auch über eine Telefonleitung, aus der Ferne, mit einem PC kommunizieren. Voraussetzung auf dem PC läuft ein entsprechendes Conduit.

Aber der Palm kann noch mehr. Die neueren Vertreter der Palm-Familie sind mit einem kompletten TCP/IP-Stack ausgerüstet (ab Palm III), d.h.: Der Nutzer kann mit seinem Handheld ins Internet. Dazu benötigt er einen Internet-Provider der den Palm unterstützt, ein analoges Modem und eine funktionstüchtige Telefondose (oder ein Handy mit eingebautem Modem) und natürlich die entsprechende Anwendungssoftware. Damit kann der Nutzer seine Mail empfangen, WWW-Seiten durchstöbern, WAP-Angebote nutzen oder sich an Newsgroups beteiligen. Selbst Telnet-Clients existieren für den Palm.

## 2.4 Anwendungsgebiete

Wie schon in vorherigen Abschnitten erwähnt, ist der Palm vor allen Dingen ein Organizer; mit der „eingebauten“ Software kann der Nutzer seine Termine verwalten, Kostenrechnungen aufstellen, Adressen speichern und To-Do-Listen eingeben. Sobald der Palm aber über die Organizerfunktionalität hinaus genutzt werden soll, wird eine Aufrüstung mit entsprechender Software oder Hardware erforderlich.

Hier ein Überblick über die weiteren Anwendungsgebiete (siehe Abbildung 4):

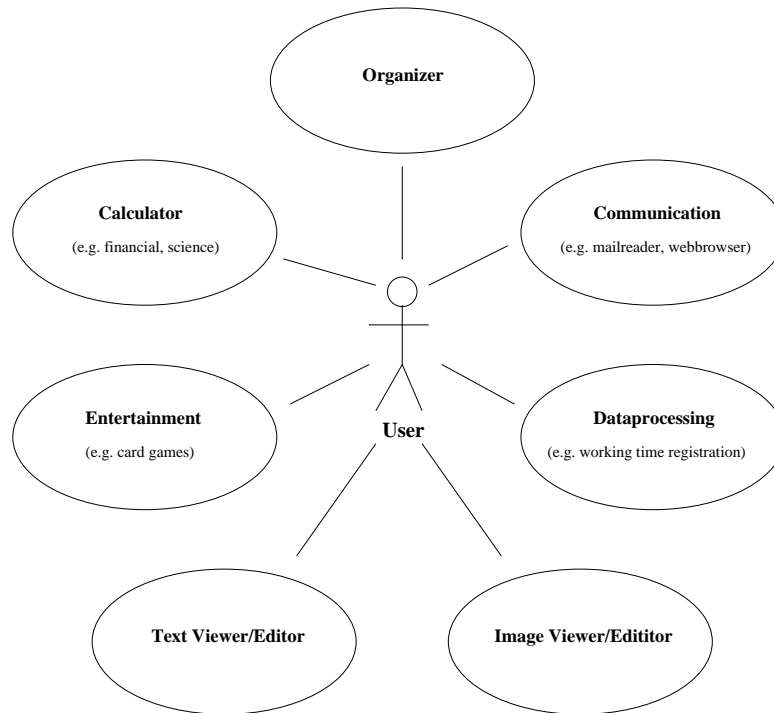


Abbildung 4: Die Anwendungsgebiete

**Calculator:** Viel mehr als die Grundrechenarten beherrscht die mitgelieferte Taschenrechner-Applikation nicht. Im Internet dagegen findet man ein großes Angebot an komplexen und wissenschaftlichen Rechenprogrammen.

**Text Viewer/Editor:** Damit sind alle Programme gemeint, mit denen man sich Textdokumente anzeigen lassen kann (z.B. eBook-Reader) oder die es einem erlauben selber Texte zu schreiben, zu editieren und zu speichern (Wordprocessing). Der Palm ist für diesen Zweck nur eingeschränkt zu gebrauchen, da er erstens ein relativ kleines Display besitzt und zweitens die Eingabe von Text mit dem Stift auf dem Eingabefeld sehr zeitaufwendig und mühsam ist. Das mag bei kleineren Notizen noch gut funktionieren, aber die Eingabe von sehr umfangreichen Texten ist damit fast unmöglich; jeder Buchstabe muss einzeln mit einer gewissen Sorgfalt gezeichnet werden. Abhilfe schafft da höchstens ein zusätzliches Keyboard, welches von Dritt-Herstellern angeboten wird. Ein weiteres Manko ist der begrenzte Speicherplatz. Nicht alle zur Zeit

existierenden Palms besitzen 4MB oder gar 8MB, wahrscheinlicher sind da eher 512KB bis 2MB<sup>10</sup>, wovon ein großer Teil mit Programmen und Daten belegt sein dürfte. Nichtsdestotrotz existiert eine große Anzahl von Dokumenten und kompletten Büchern im Internet, die im Format einer Palm-Datenbank für spezielle Textreader gespeichert sind. Auf verschiedene Schriftarten oder -größen muss hier aber weitestgehend verzichtet werden.

**Image Viewer/Editor:** Selbst für den Palm gibt es Bildbetrachtungssoftware und einfache Mal-Programme, mit den bestehenden Einschränkungen an Bildschirmgröße und Speicherkapazität.

**Entertainment:** Damit ist vor allen die große Anzahl von Spielen, die für den Palm existieren, gemeint. Die Bandbreite reicht von Karten-Spielen, Geschicklichkeits-Spielen und Strategie-Spielen, bis hin zu Aktion-Titeln.

**Communication:** Die wesentlichen Kommunikationsprogramme wurden schon genannt (siehe Abschnitt 2.3, auf Seite 8): Mailprogramme, Newsreader, WWW- und WAP-Browser, Telnet-Clients, Programme zum Datenaustausch usw.

**Dataprocessing:** Alle Programme die zur Erfassung, Verwaltung, Speicherung und Präsentation von Daten dienen, gehören in diese Kategorie. Das sind z.B. Tabellenkalkulationen, Programme zur Arbeitszeiterfassung, Programme die GPS-Signale aus einem GPS-Empfänger interpretieren und anzeigen können sowie Datenbank-Applikationen.

Im Endeffekt erschließen sich dem Palm damit generell alle Anwendung eines normalen PCs.

## 2.5 Die Palm Programmierung

Die meisten Programme für den Palm werden wohl in "C" mit der kommerziellen Entwicklungsumgebung *Codewarrior* von *Metrowerks*<sup>11</sup> oder mit dem für fast alle Systeme vorhandenem freien GNU-Pilot-SDK geschrieben, aber auch die Sprachen Assembler, Basic oder Forth können genutzt werden. Zur

---

<sup>10</sup>Der aktuelle Einsteiger-Palm M100 wird beispielsweise mit 2MB ausgeliefert.

<sup>11</sup><http://www.metrowerks.com>

Zeit existieren sogar die Bestrebung (unter anderem von SUN) eine Java Virtuell Maschine für den Palm umzusetzen, welche aber spezielle Java-Klassen zur Programmierung verwendet.

Palm-Applikationen werden gewöhnlicherweise auf einem Desktop-System entwickelt und mit einem entsprechenden Emulator (POSE, xcopilot) getestet bevor sie auf einen Palm aufgespielt und dort auch genutzt werden können. Am Beispiel des GNU-Pilot-SDK soll verdeutlicht werden, wie Pro-

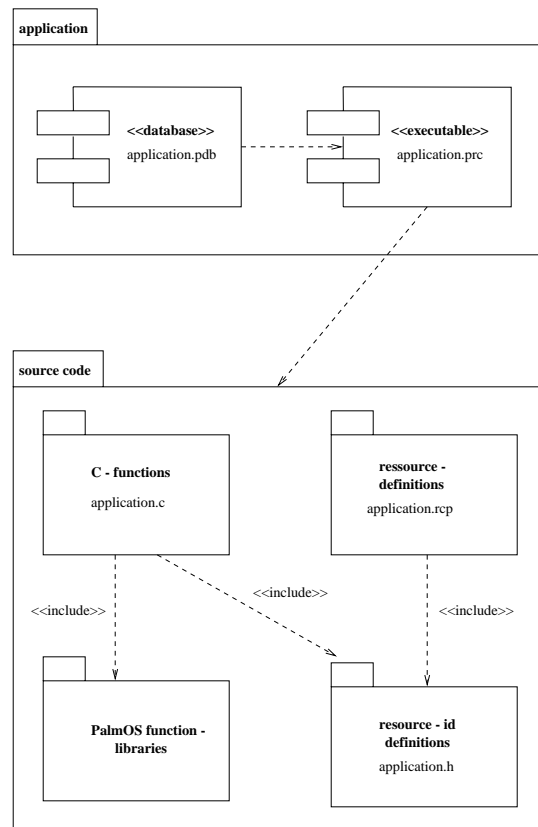


Abbildung 5: Die Source-Files für das GNU-Pilot-SDK

gramme für den Palm entwickelt werden. Die Abbildung 5 zeigt, welche Source-Files benötigt werden, damit das SDK eine lauffähige Applikation (*application.prc*) generieren kann.

Da wäre als erstes der "C"-Quellcode zu nennen (*application.c*). Palm Programme sind Event-Driven, d.h., dass die Applikation nur Aktionen aufgrund eines Ereignisses ausführen kann. Solche Ereignisse (Events) sind beispielsweise der Start eines Programms, das Berühren des Bildschirms mit

dem Stift oder das Betätigen eines Buttons. Die Auswertung dieser Ereignisse übernimmt dann der entsprechende Event-Handler. Beim Start eines Programms könnten z.B. globale Variablen initialisiert oder die Datenbank geöffnet werden. Mit der Betätigung eines Buttons dagegen könnte der Nutzer beispielsweise seine Eingabe bestätigen oder eine vom Programm gestellte Frage beantworten.

Das nächste File (*application.rcp*) enthält die Definition aller im Programm vorkommenden Ressourcen. Solche Ressourcen sind z.B. die verschiedenen Bildschirme der Applikation (*FORMS*), die darin enthaltenen Bildelemente (z.B.: *Buttons*, *Tables*, *Entry Fields* und *Scrollbars*), Menüs, Alert-Dialoge und Fontdefinitionen.

Alle Ressourcen besitzen eine Nummer, mit der sie eindeutig identifiziert werden können. Diese Nummer wird im C-Quellcode zur Auswahl der entsprechenden Ressource genutzt. Um die Lesbarkeit des Quelltextes zu erhöhen, kann der Programmierer die Ressourcen auch mit einem Namen versehen, muss dann diesem Namen aber wiederum eine eindeutige Nummer zuweisen. Dies kann er entweder im Definitionsteil des C-Quellcodes und des Resource-Files tun oder aber er schreibt die Zuweisungen in das Resource-ID-File (*application.h*). Dieses File wird in das Resource-File und in den "C"-Quelltext "inkludiert". Jetzt kann der Programmierer die Ressourcen über ihren Namen ansprechen.

Des Weiteren werden zum Erstellen von Palmapplikationen noch Funktions-Bibliotheken des PalmOS benötigt. Diese Header-Files enthalten die Namen von Systemfunktionen und vordefinierten zusammengesetzten Variablen. Beispiele dafür sind Funktionen zum Zugriff auf Bildelemente, zur Nutzung des Memory-Managers und zum Zugriff auf die Applikations-Datenbank (*application.pdb*). Die Abbildung 6 zeigt, welche Tools des GNU-Pilot-SDK benutzt werden, um aus dem Sourcecode und den Library-Files eine lauffähige Applikation zu generieren. Das vom M68K-Palm-OS-GCC aus dem "C"-Quellcode und den inkludierten Library-Files erzeugte Objekt-File und die vom Resource-Compiler "PILRC" aus dem Resource-File erzeugten Dateien, werden von "OBJ-RES" zum entgeltigen Programm zusammengelinkt, welches dann entweder auf dem Palm oder dem Palm-Emulator ausgeführt werden kann.

Die von einer Applikation erzeugten Palmdatenbanken sind recordbasiert aufgebaut. Damit ist gemeint, dass Programme auf die einzelnen Einträge (Records) der Datenbanken, nur über einen Positionsindex zugreifen können. Die zu diesem Zweck vorhandenen Funktionen des Palm-Betriebssystems liefern auf Anfrage ein Handle auf den entsprechenden Record zurück, mit dem ein

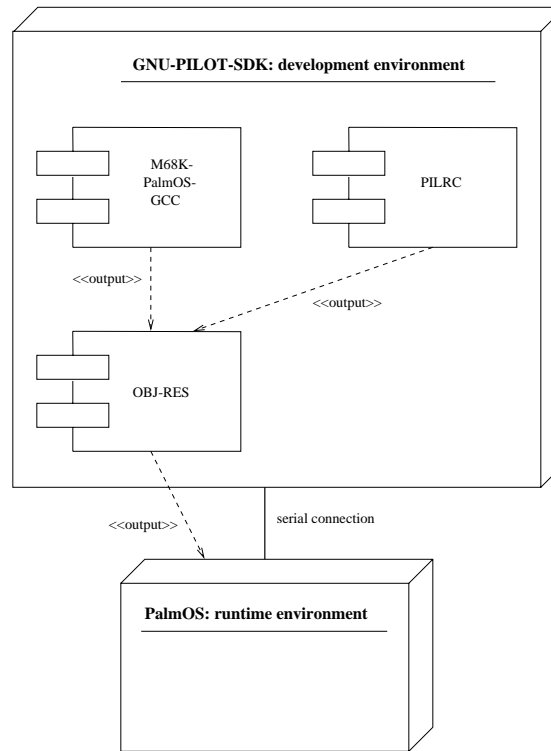


Abbildung 6: Das GNU-Pilot-SDK

Pointer vom Typ des Records in der Datenbank generiert werden kann. Somit hat der Programmierer die Möglichkeit zum Zugriff auf jedes einzelne Feld des Eintrages. Zusätzlich ist noch zu sagen, dass Palmdatenbanken nicht auf einen Recordtyp beschränkt sind. Jeder Record in der Datenbank kann anders aufgebaut sein.

Einen kompletten Einstieg in die Palmprogrammierung mit Codewarrior oder GCC bietet [RMS99].

## 2.6 Zusammenfassung

Die Auswertung, der in diesem Kapitel vorgestellten technologischen Grundlagen des Palm-Handheldcomputer, ergibt folgende Resultate:

- Der Palm kann dem Betrieb von Programmen jeglicher Art dienen, mit Einschränkungen beim Eingabevolumen, bei den Darstellungsmöglichkeiten und beim Hauptspeicherbedarf.

- Er kann als mobiler Speicher ausgewählter Daten für autonome Anwendungen genutzt werden.
- Durch die Infrarotschnittstelle ist ein einfacher Datenaustausch zwischen Palmcomputern für die verteilte Informationsnutzung möglich.
- Er ist gut zur (evtl. verteilten) mobilen Datenerfassung geeignet. Dabei auch als Frontend für stationäre Software-Systeme.
- Prinzipiell ist eine komplette Internetanbindung mit Nutzung von WWW-Inhalten möglich, allerdings mit Einschränkungen durch die Bildschirmgröße.

Die Nutzung des Palm-Computers, als mobiles Frontend für bestehende oder zu entwerfende stationäre Softwaresysteme, ist nicht Gegenstand dieser Diplomarbeit. Das Ziel ist der Entwurf von autonomen, komplett auf dem Palm ablauffähigen CASE-Werkzeugen.



## 3 CASE-Tools im Software-Lebenszyklus

Dieses Kapitel untersucht den Einsatz von CASE-Tools im Lebenszyklus eines Softwareprojektes und überprüft die Möglichkeit und den Sinn einer Umsetzung für den Palm-Computer.

### 3.1 Die Software-Entwicklung, -Anwendung und -Wartung

#### 3.1.1 Die Problemdefinition

In der Phase der Problemdefinition geht es um: „*Die Beschreibung der Anforderungen an ein zu erstellendes Software-Produkt bzw. Software-System.*“ [Dum00, ab Seite 23].

*Aktivitäten in der Phase der Problemdefinition:*

- Erstellung des Lasten- oder Pflichtenheftes durch den Auftraggeber, in dem alle Anforderungen an das zu erstellende Software-Produkt festgehalten sind,
- Erste Überprüfung der Anforderungen in der formulierten Aufgabenstellung durch den Auftragnehmer,
- Besprechung der Produktidee zwischen Auftraggeber und Auftragnehmer (z.B. durch Brainstorming),
- Erstellung projektbezogener Support-Dokumente, wie z.B.:
  - ein Fachbegriffskatalog,
  - eine Gesetzessammlung,
  - eine Expertenliste,
  - eine Projektstatistik und
  - eine Trendanalyse

*Verwendete CASE-Tools:*

- Textverarbeitungsprogramme, zum Schreiben der Anforderungs- und Supportdokumente (z.B.: *MS Word*<sup>12</sup>),
- Grafik-Programme, um die anfallenden Dokumente mit Skizzen, Bildern oder Diagrammen zu ergänzen (z.B.: *MS Visio*, *MS Excel*),
- Videokonferenzsysteme (z.B.: *MBone*<sup>13</sup>), für die verteilte Kommunikation (Brainstorming zwischen Mitgliedern von verschiedenen Orten aus)

*Fazit für die Palm-Anwendung:*

Der Palm kann zur Erstellung des Pflichtenheftes und der Support-Dokumente kaum genutzt werden. Sinnvoller ist hier der Einsatz herkömmlicher Desktop-Systeme. Der Vorteil des Palms - seine Mobilität - kann dagegen bei den Besprechungen (Brainstorming) zwischen Auftraggeber und Auftragnehmer zum Einsatz kommen. Die typischen Organizer Applikationen, wie der *Merkzettel-Editor* und die *Aufgaben-Liste (To-Do-List)*, können hier als Gedankenstütze dienen. Die dafür notwendige Schreibearbeit wird am besten schon vorher am PC mit *Palm-Desktop* oder einem entsprechendem Programm erledigt.

Um die Informationen der erstellten Support-Dokumente immer verfügbar zu haben, bieten sich palmbasiert Datenbanksysteme (wie z.B. *ThinkDB*<sup>14</sup>) an. Damit könnten u.a. der Fachbegriffskatalog, die Gesetzessammlung und die Expertenliste auch unterwegs zur Verfügung stehen.

### 3.1.2 Die Anforderungsanalyse

In der Phase der Anforderungsanalyse werden die in dem Pflichtenheft stehenden Anforderung hinsichtlich ihrer Korrektheit, Vollständigkeit, Sachgerechtigkeit, Konsistenz und Realisierbarkeit überprüft (siehe [Dum00, ab Seite 32]).

*Aktivitäten in der Phase der Anforderungsanalyse:*

---

<sup>12</sup><http://www.microsoft.com>

<sup>13</sup><http://www.mbone.de>

<sup>14</sup><http://www.thinkingbytes.com/>

- die fachspezifische Begriffskontrolle, zur Überprüfung der korrekten Anwendung von Fachbegriffen,
- die allgemeine Konsistenzkontrolle der aufgelisteten Anforderungen,
- die Analogiemethode, zur Überprüfung der Realisierbarkeit der Anforderungen durch Vergleich mit bestehenden Entwicklungserfahrungen,
- die Interview-Technik (siehe [PJ91, Seite 46]), zur Einbeziehung des Auftraggebers zur Klärung oder Richtigstellung von Anforderungen,
- Recherche nach weiteren externen und internen Informationen, zur Überprüfung der Anforderungen

*Verwendete CASE-Tools:*

- Recherche Tools, wie WWW-Browser (z.B.: *Netscape*<sup>15</sup>) und Suchmaschinen,
- Textbetrachtungsprogramme, wie Postskript-Viewer oder PDF-Viewer (z.B.: der *Acrobat Reader*<sup>16</sup>), um die Dokumente der Phase der Problemdefinition anzuzeigen,
- Textverarbeitungsprogramme, um die gewonnenen Erkenntnisse aus den Analysen aufzuschreiben,
- Grafik-Programme, um die anfallenden Dokumente mit Skizzen, Bildern oder Diagrammen zu ergänzen.
- Spezielle Tools, die das Abspeichern und Tracen von Anforderungen bis hin zum fertigen Softwaremodul und zur Dokumentation ermöglichen (siehe [Lew91]: Chapter 5, Software Requirements Specification),
- Programmiersysteme, zur Erstellung von Prototypen

*Fazit für die Palm-Anwendung:*

Auch in dieser Phase bestimmen hauptsächlich Textverarbeitungs- und Textbetrachtungsprogramme das Arbeitsumfeld des beteiligten Mitarbeiter. Die Umsetzung der sogenannten Trace-Tools, zur Verwaltung und Verfolgung

---

<sup>15</sup><http://www.netscape.com>

<sup>16</sup><http://www.adobe.de>

aller Anforderungen über den kompletten Lebenszyklus hinweg, ist kaum denkbar, da die Datenmenge zu groß ist und die Beziehungen zwischen den Informationsblöcken zu komplex sind.

Prinzipiell möglich mit dem Palm ist dagegen die mobile Recherche im Internet, zur Unterstützung der Analyse der Anforderungen, wenn die gegebenen Einschränkungen einen solchen Einsatz nicht verhindern. Besonders gut geeignet ist in diesem Zusammenhang das immer größer werdende WAP-Angebot, welches speziell für die kleinen Bildschirme von elektronischen Organismen und Handys entwickelt wurde.

### 3.1.3 Die Spezifikation

In der Phase der Spezifikation werden die Anforderungen in ein Modell umgesetzt, welches die Funktionalität des zu entwickelnden Software-Produktes vollständig beschreibt (siehe [Dum00, ab Seite 37]).

*Aktivitäten in der Phase der Spezifikation:*

- Erstellung eines projektspezifischen Wörterbuches (Data Dictionary) oder eines Repositories,
- Aufwandschätzung und Ermittlung der Komplexität der Komponenten des Systems, z.B. nach dem CoCoMo-Modell von Boehm [Tha97, Seite 230ff und Seite 270ff] oder der Function-Points-Methode<sup>17</sup>,
- Anfertigung einer Machbarkeitsanalyse, anhand von Simulationen oder der Erstellung von Prototypen,
- Modellierung des Software-Systems (z.B.: mit UML [EP98, SvG00])

*Verwendete CASE-Tools:*

- Textverarbeitungsprogramme, zur Niederschrift der Spezifikationen,
- Programme zur Erstellung, Abspeicherung und Präsentation des Data Dictionary oder des Repository,
- Programme zur Berechnung der Komplexität und zur Aufwandsschätzung (z.B.: *CoCoPro* [Lew91, Seite 47], *Function Point Execution Example*<sup>18</sup>),

---

<sup>17</sup><http://www.ifpug.com>

<sup>18</sup><http://ivs.cs.uni-magdeburg.de/sw-eng/us/java/fp/>

- Grafik-Programme, um Diagramme und Modelle zu erstellen,
- Spezielle Modellierungs-Programme (z.B.: *Rational Rose*<sup>19</sup> von den UML Erfindern)

*Fazit für die Palm-Anwendung:*

In dieser Phase existieren mehrere Möglichkeiten den Palm zur Unterstützung des Spezifikations-Prozesses einzusetzen. Sicherlich kann er auch hier nicht die Textverarbeitung oder das Grafik-Programm ersetzen und auch die Modellierung sollte besser mit einem Desktop-Computer erledigt werden. Allerdings könnte die ständige Verfügbarkeit des Palms gerade in diesem Zusammenhang sehr von Vorteil sein, wenn man bedenkt, dass einem oft unterwegs (z.B. im Zug) die besten Ideen kommen. In diesem Fall ist ein Ausweichen auf den Merktzeteeditor oder auf frei verfügbare Zeichenprogramme denkbar. Des weiteren ist es ist möglich, den Palm zur manuellen Komplexitäts- oder Aufwandsberechnung einzusetzen (z.B. als Function-Point Zähltool).

### 3.1.4 Der Entwurf

Die Phase des Entwurfs dient zur Erstellung der Architektur des Software-Produktes, d.h.: Die software- und hardwarebezogene Struktur des Systems wird entwickelt. Dazu gehören die Komponenten des Systems, die Schnittstellen und die Beziehungen zwischen den Komponenten (siehe [Dum00, ab Seite 47]).

*Aktivitäten in der Phase des Entwurfs:*

- Erstellung der allgemeinen Architektur, durch Nennung der einzelnen Komponenten und ihrer Beziehungen untereinander,
- Beschreibung der einzelnen Komponenten durch die Zusammenfassung der zugehörigen Funktionalitäten,
- Recherche, Überprüfung und Beschaffung von Software, die in dem zu entwickelndem Softwaresystem genutzt werden soll (Akquisition),
- Erstellung einer Konfiguration, d.h. einer Variante der Grundarchitektur des Softwareproduktes

---

<sup>19</sup><http://www.rational.com>

*Verwendete CASE-Tools:*

- Textverarbeitungsprogramme, zur Beschreibung einzelner Punkte des Software-Entwurfs,
- Grafikprogramme, um Grafiken, Modelle und Diagramme zu erstellen,
- spezielle CASE-Tools die den Entwurf teilweise oder komplett unterstützen (z.B.: UML-Tools),
- Webbrowser, zur Recherche im Internet für die Akquisition.
- Konfigurationsmanagement-Programme (z.B.: *QVCS*)

*Fazit für die Palm-Anwendung:*

Es gibt eine Reihe von CASE-Tools, die den Softwareentwurf unterstützen, wie z.B. *Rational Rose* von den UML-Erfindern (siehe Abschnitt 3.1.3). Diese Tools ermöglichen u.a. die grafische Eingabe der Modelle, deren Konsistenzprüfung und teilweise die automatische Quellcode-Generierung. Der Palm ist aufgrund seines kleinen Displays und seiner standardmäßig fehlenden Tastatur für solche Tätigkeiten nur eingeschränkt oder gar nicht nutzbar.

Wer die Internetrecherche mit seinem Palm machen möchte, benötigt dazu die entsprechende Software (Webbrowser) und Hardware (Modem, Handy mit eingebautem Modem) sowie einen Internet-Provider, der den Palm unterstützt. Allerdings schränkt auch hier das kleine Display die Nutzung sehr stark ein, z.B. ist die Darstellung von aktiven Inhalten von WWW-Seiten (z.B. Java) nicht oder noch nicht möglich.

### 3.1.5 Die Implementation

In der Phase der Implementierung werden die Ergebnisse des Entwurfs in ablauffähige Software umgesetzt (siehe [Dum00, ab Seite 63]).

*Aktivitäten in der Phase der Implementation:*

- Erzeugung des Quellcodes, durch manuelles Editieren oder durch automatische Quellcodegenerierung aus bestehenden Modellen,
- Testen der erstellten Softwaremodule,
- Integration der Softwaremodule in das Softwaresystem,

- Fertigstellung der Nutzer- und Entwicklerdokumentation,
- Installation der Software, d.h.: Erstellung eines Installationsprogramms und Zusammenstellung des Softwaresystems auf elektronischen Datenträgern (z.B.: Diskette oder CD-Rom)

*Verwendete CASE-Tools:*

- Integrierte Entwicklungsumgebungen (z.B.: *JBuilder* von Inprise<sup>20</sup>),
- Editoren, zum Aufschreiben des Quellcodes,
- Compiler und Linker zum Übersetzen des Quellcodes in ablauffähige Software,
- Debugger, zur Fehlersuche in Software,
- Emulatoren, falls die Zielplattform der Software sich von der Entwicklungsplattform unterscheidet (Beispiel Palm-Programmierung),
- Quellcode Generatoren (z.B.: *Select-Tool*, für die Umsetzung von UML-Modellen in Quellcode),
- Dokumentationsgeneratoren und Textverarbeitungsprogramme, zur Erstellung der Dokumentationen,
- Quellcode Mess- und Analysetools (z.B.: *Logiscope*<sup>21</sup>),
- Benchmark-Programme

*Fazit für die Palm-Anwendung:*

Aufgrund beschränkter Hard- und Softwareressourcen ist der Palm im Prozess der Softwareprogrammierung kaum einsetzbar, selbst Palm-Applikationen werden größtenteils auf anderen Computersystemen entwickelt. Allerdings könnte man ihn beim Softwaretest einsetzen: Zur Erstellung und Verwaltung von Checklisten, zum Ausfüllen von Fehler-Protokollen und zur Unterstützung von Reviews und Audits, unter Ausnutzung des Merktzetteleditors und der Aufgabenliste. Ein weiteres Einsatzgebiet ist die Unterstützung der Berechnung und manuellen Zählung von Softwaremaßen.

---

<sup>20</sup><http://www.inprise.com>

<sup>21</sup><http://ivs.cs.uni-magdeburg.de/sw-eng/us/CAME/CAME.tools.logiscope.shtm>

Der Nutzer gibt die Teilergebnisse seiner Zählungen in eine spezielle Palm-Applikation ein, die speichert sie ab und kann dann damit weitere Berechnungen anstellen oder mit vorhandenen Ergebnissen vergleichen. Mit Einschränkungen ist auch eine tabellarische oder grafische Darstellung der Daten möglich. Diese Anwendungsmöglichkeit ist nicht nur auf die Implementation beschränkt. Sie kann in allen Phasen, wo Softwaremaße genutzt werden, zum Einsatz kommen.

### 3.1.6 Die Erprobung

In der Phase der Erprobung wird das fertige Softwareprodukt unter realen Anwendungsbedingungen getestet (siehe [Dum00, ab Seite 93]).

*Aktivitäten in der Phase der Erprobung:*

- Erarbeitung eines Abnahmekonzeptes, welches u.a. die benötigte Software und Hardware, die personellen Ressourcen und das nötige Änderungsbudget umfasst,
- Erstellung einer Mängelliste durch den Auftragnehmer,
- Erstellung eines Bearbeitungskonzeptes, zur Auslieferung der Software und ihrer Einführung beim Kunden.

*Fazit für die Palm-Anwendung:*

In dieser Phase des Lebenszyklusses kommen hauptsächlich Textverarbeitungssysteme zum Einsatz. Mit ihnen wird das Abnahmekonzept, die Mängelliste und das Bearbeitungskonzept niedergeschrieben. Einzig die palmbasierten Fehlerprotokoll-Formulare aus der Implementationsphase könnten zur Unterstützung der Erstellung der Mängelliste eine Verwendung finden.

### 3.1.7 Die Wartung

Die Softwarewartung umfasst Aktivitäten zur Erweiterung, Anpassung, Korrektur, Verbesserung und Vorbeugung an einem ausgelieferten Software-Produkt (siehe [Dum00, ab Seite 95]).

*Aktivitäten in der Phase der Wartung:*

- Erweiterung,
- Anpassung,
- Korrektur,
- Verbesserung,
- Vorbeugung.

*Verwendete CASE-Tools:*

- Quellcode Analysetools,
- Statistiktools, zur Analyse der Fehlerhäufigkeit in Softwaremodulen,
- Performance-Tools,
- Konfigurationsmanagement-Tools, zur Versionserstellung.

*Fazit für die Palm-Anwendung:*

Die CASE-Tools dieser Phase kommen auch schon in früheren Phasen vor, nur mit dem Unterschied, dass sie möglicherweise von anderen Personengruppen benutzt werden (falls z.B. die Wartungsabteilung und die Entwicklungsabteilung voneinander getrennt sind). Dem Palm erschließen sich daher keine neuen Anwendungsgebiete.

### **3.1.8 Die Anwendung**

Die Phase der Softwareanwendung ist die Nutzung des Softwareproduktes durch den Kunden (siehe [Dum00, ab Seite 101]).

*Aktivitäten in der Phase der Anwendung:*

- Erstellung eines Einführungs- und Nutzungskonzeptes,
- Einführung des Softwareproduktes,
- Nutzung des Softwareproduktes,

- Umstellung auf neue Versionen,
- Ablösung durch ein anderes Softwareprodukt.

*Verwendete CASE-Tools:*

Zur Erstellung des Einführungs- und Nutzungskonzeptes kommen Textverarbeitungssysteme, Grafikprogramme und Modellierungstools zum Einsatz.

*Fazit für die Palm-Anwendung:*

Die Mobilität und ständige Verfügbarkeit des Palms kann bei dieser Phase genutzt werden, um die gesammelten Informationen und Erfahrungen der Anwendung eines Softwareproduktes sowie Teile des Einführungs- und Nutzungskonzeptes immer verfügbar zu haben. Dazu zählt beispielsweise die Knowledgebase der Anwendung oder der Organisationsplan aus dem Einführungskonzept.

## 3.2 Projekt begleitende CASE-Tools

Neben den CASE-Tools, die die einzelnen Prozesse innerhalb der Phasen des Softwarelebenszyklusses unterstützen, existieren Programme zur Begleitung des kompletten Projektverlaufes.

### 3.2.1 Projektmanagement-Programme

Zum Projektmanagement gehören laut [Mah92] folgende Aufgaben:

**Die Projektplanung** Dieser Schritt beinhaltet die Zerteilung des Projektes in Teilaufgaben oder Prozesse, die Verwaltung und Zuordnung der Ressourcen zu den einzelnen Prozessen und die Abschätzung der benötigten Zeit. Das Ergebnis ist ein Projektplan.

**Die Projektkontrolle** Wurde der Projektplan erstellt, besteht die Hauptaufgabe in der Kontrolle des Projektverlaufes. Es muss der Fortschritt des Projektes überwacht und gegebenenfalls der Projektplan an die Gegebenheiten angepasst werden. Dies kann eintreten, wenn z.B. Ressourcen ausfallen, Zeitvorgaben nicht eingehalten werden konnten oder neue Teilaufgaben anfallen.

**Die Auswertung des Projektes** Nach Abschluss des Projektes ist eine Auswertung erforderlich. Dort wird überprüft, wie die einzelnen Vorgaben hinsichtlich der benötigten Zeit und des Verbrauches an Ressourcen eingehalten wurden. Dies so gewonnen Ergebnisse können als Erfahrungen in die Planung kommender Projekte einfließen.

Zur Beschreibung und Präsentation von Projektplänen dient die Netzplantechnik. Mit ihr können die einzelnen Aktivitäten oder Vorgänge eines Projektes und ihre Beziehungen untereinander dargestellt werden. Abbildung 7 vergleicht die gängigsten drei Netzplanarten.

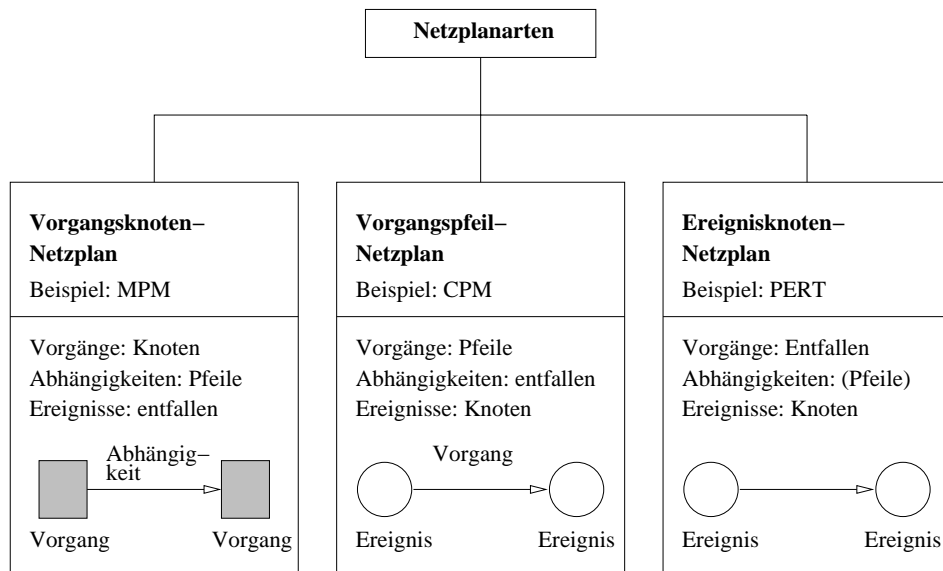


Abbildung 7: Netzplanarten nach [Bal98]

Ein Programm, das das Projektmanagement unterstützen will, sollte folgenden Funktionen beinhalten:

- Erstellung und Verwaltung von Projekten,
- Definierung von Vorgängen,
- Festlegung von Beziehungen zwischen den Vorgängen,
- Erstellung und Verwaltung von Ressourcen,

- Manipulation und automatische Neuberechnung des Projektkalenders,
- Aufbereitung der Informationen in Netzplänen, Balkendiagrammen und Tabellen,
- Import und Export von Daten.

Zwei Beispiele für Projektmanagement-Programme sind *MicroPlaner+*<sup>TM</sup> von Micro Planning International (siehe [Lew91] Abschnitt "3. The Project") und *MS Project* (siehe [Mah92]).

*Fazit für die Palm-Anwendung:*

Die Umsetzung eines Projektmanagementprogramms für palmbasierte Computer ist durchaus denkbar, wenn eine Konzentrierung der Daten auf das Wesentliche und eine durch die Größe des Displays bedingte Einschränkung der Darstellung der Netzpläne beachtet wird. Tatsächlich existieren eine Reihe von mehr oder weniger komplexen Projektmanagement-Programmen für den Palm. Beispiele hierfür sind das freie, unter der GPL stehende, *Project Manager*<sup>22</sup> und *Project Punch*<sup>23</sup> (siehe Abbildung 8).

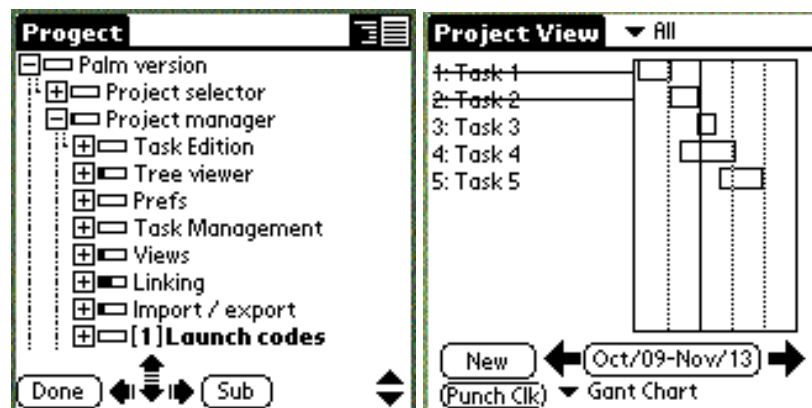


Abbildung 8: Screenshots von *Project Plan* (l.) und *Project Punch* (r.)

Hinzugefügt werden muss noch, dass Projektmanagement-Programme nicht nur auf die Erstellung von Software beschränkt sind. Sie können bei aller Art von Projekten eingesetzt werden, bei denen eine computergestützte Projektplanung erforderlich oder sinnvoll ist.

<sup>22</sup><http://project.sourceforge.net>

<sup>23</sup><http://www.geocities.com/projpunch/>

### 3.2.2 Programme zur Qualitätssicherung

Es gibt verschiedene Methoden, um die Qualität des Softwareentwicklungsprozesses zu erhöhen. Beispiele dafür sind die Zertifizierung nach der *ISO 9000 Norm*<sup>24</sup> (siehe [OG97]) oder die Bewertung nach dem *Capability Maturity Model (CMM)*<sup>25</sup> (siehe [Tha97, ab Seite 48]):

**ISO 9000** Laut [Bal98, Abschnitt 4.1], legt das ISO 9000-Normenwerk für das Auftraggeber-Lieferanten-Verhältnis einen allgemeinen, übergeordneten, organisatorischen Rahmen zur Qualitätssicherung von materiellen und immateriellen Produkten fest. Dabei ist für die Software-Qualitätssicherung die ISO 9001 relevant:

*„Allgemeine Einführung und Überblick über den Zusammenhang der Normen dieser Serie. Leitfaden zur Auswahl und Anwendung dieser Normen in bezug auf das Qualitätsmanagement, die Elemente dieses Qualitätsmanagementsystems und der Qualitätssicherungsstufe.“*

Für den Softwareentwicklungsprozess existiert eine Erweiterung der ISO 9001, die ISO 9000-3, mit dem Titel: *„Normen zum Qualitätsmanagement und zur Qualitätssicherung – Teil 3 – Leitfaden für die Anwendung von ISO 9001 auf die Entwicklung, Lieferung und Wartung von Software.“*

Die Überprüfung eines Prozesses hinsichtlich seiner ISO Konformität geschieht anhand eines Fragebogens, welcher 155 Fragen zu folgenden Themen (siehe [Dum00, Seite 201]) enthält:

- Konfigurationsmanagement,
- Dokumentationskontrollen,
- Qualitätsreports (als Reviews oder Audits),
- Qualitätsplänen (als Testpläne, Abnahmepläne usw.),
- Softwagemessung in seiner prinzipiellen Form,
- Anforderungsgerechte Soft- und Hardwarebeschaffung,
- Einsatz von CASE-Tools und moderner Entwicklungstechnologien,
- Kosten- und Aufwandsanalysen,

---

<sup>24</sup><http://www.iso.com>

<sup>25</sup><http://ivs.cs.uni-magdeburg.de/sw-eng/us/java/CMM/index.js.shtml>

- Schulungen und Entwicklungspersonal,
- Nutzergerechte Softwarewartung.

Dabei gilt ein Prozess als ISO-9000 zertifiziert, wenn alle Fragen positiv beantwortet werden konnten.

**CMM** Im Unterschied zur ISO 9000-Norm ist das Capability Maturity Model direkt auf die Softwareentwicklung hin ausgerichtet. Als Ergebnis der Bewertung wird der Prozess in fünf Reifegradstufen mit folgender Spezifikation eingeteilt (siehe [Dum00, Seite 203]):

1. **Initial** Abhängig von einzelnen Personen und deren Fähigkeiten
2. **Repeatable** Software-Projektmanagement, Software-Qualitätssicherung, Konfigurationsmanagement, Kosten-/Aufwandschätzungen
3. **Defined** Prozessdefinition auf der Grundlage einer Prozessanalyse, integriertes Softwaremanagement, CASE-Tool-Anwendung, Schulungsprogramm, Peer-to-Peer Reviews
4. **Managed** Quantitatives Prozessmanagement auf der Grundlage von Metriken, Software-Qualitätsmanagement
5. **Optimizing** Prozessänderungsbeherrschung, gezielte Fehlervermeidung, Technologieänderungsbeherrschung.

Auch hier wird die Einteilung anhand eines Fragenkataloges vorgenommen, mit den Schwerpunkten Organisation, Projektmanagement, Prozessmanagement und Technologie. Die Fragen sind mit der Reifegradstufe gekennzeichnet, zu welcher ihre Beantwortung mit "Ja" führt. Um eine bestimmte Stufe erreichen zu können, müssen aber natürlich auch alle Fragen der niedrigeren Levels positiv beantwortet werden.

*Fazit für die Palm-Anwendung:*

Ein Handheldcomputer, wie der Palm, ist aufgrund seiner Mobilität sehr gut zur Unterstützung des Zertifizierungsprozesses geeignet, da der Zertifizierer die gestellten Fragen nur durch einen Dialog, mit den an dem Softwareentwicklungsprozess beteiligten Personen, beantworten kann. Ein entsprechendes Programm sollte die Fragen den Kategorien bzw. Stufen nach anzeigen und das Ergebnis projektbezogen abspeichern und präsentieren können.

### 3.3 CASE-Umgebung

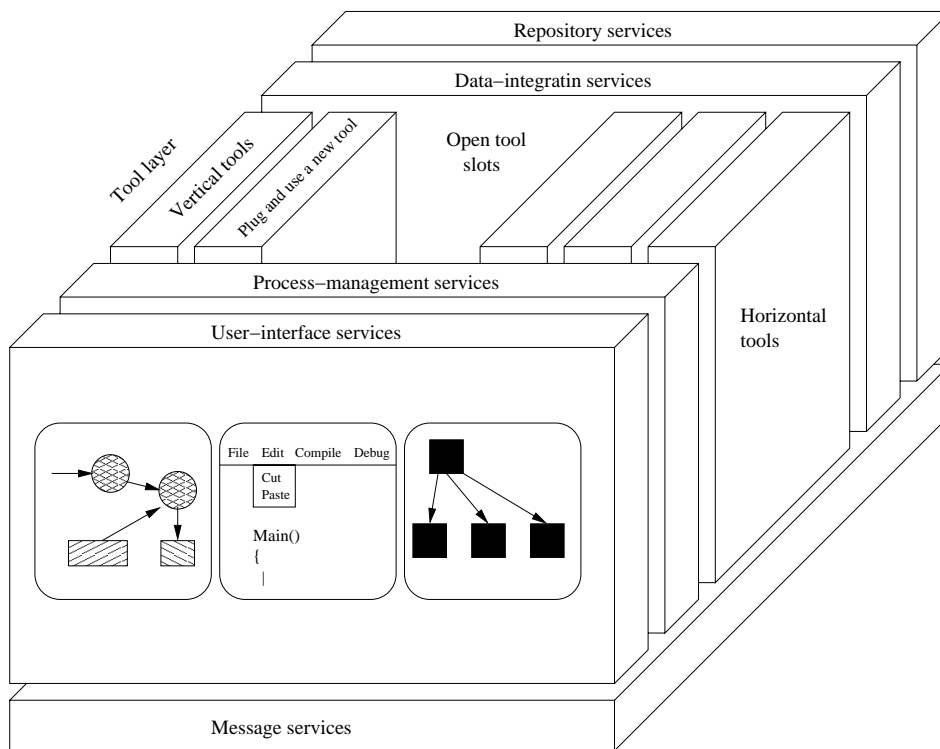


Abbildung 9: NIST/ECMA Referenzmodell einer CASE-Umgebung, siehe [Pff99]

Eine CASE-Umgebung besteht aus einer CASE-Plattform und einer Anzahl von CASE-Tools. Die CASE-Plattform bietet den verschiedenen CASE-Tools einen gemeinsamen Rahmen, welcher definierte Schnittstellen und Dienste, wie z.B. ein Repository, ein Message Service und ein einheitliches Nutzerinterface enthält (siehe auch [Bal98, Seite 592ff]). In Abbildung 9 ist das NIST<sup>26</sup>/ECMA<sup>27</sup> Referenzmodell einer solchen CASE-Umgebung zu sehen. Die Dienste (Services) des Modells bilden die CASE-Plattform, in die die CASE-Tools eingebettet sind. Die "vertikalen Tools" repräsentieren die Werkzeuge, die den gesamten Lebenszyklus begleiten und die "horizontalen

<sup>26</sup><http://www.nist.gov>

<sup>27</sup><http://www.ecma.ch>

Tools“ die Werkzeuge, die der Unterstützung der einzelnen Phasen dienen. Durch die Integrierungsfähigkeit und die Offenheit der Schnittstellen soll garantiert werden, dass externe Werkzeuge Daten mit der Plattform austauschen und dass Nutzer und Drittanbieter eigene Werkzeuge integrieren können.

*Fazit für die Palm-Anwendung:*

Eine CASE-Umgebung für den Palm zu schreiben ist wenig sinnvoll, da nur ein Ausschnitt aller notwendigen CASE-Tools überhaupt auf dem Palm eine Einsatzmöglichkeit finden (siehe Abschnitt 3.1 und Abschnitt 3.2). Trotzdem könnte es Vorteile bringen, für die möglichen CASE-Tools eine gemeinsame Plattform zu schaffen, welche erstens mit einer gemeinsamen Datenbasis ausgestattet ist, um Redundanzen in der Eingabe und Speicherung von Informationen zu vermeiden und zweitens mit definierten Schnittstellen, um den Datenaustausch mit externen Programmen (z.B. auf Desktopcomputern) zu ermöglichen.

Ein Beispiel für eine CASE-Umgebung ist *case/4/0* der deutschen Firma *microTOOL*<sup>28</sup>

### 3.4 Zusammenfassung

Die Überprüfung des Software-Lebenszyklusses in diesem Kapitel zeigt schon ziemlich deutlich, in welchen Fällen der Einsatz eines Palm-Computers möglich und sinnvoll ist. Seine größten Stärken kann der Palm ausspielen, wenn es darum geht seine sowieso vorhandenen Eigenschaften als Organizer auszunutzen (z.B. durch die Verwendung der eingebauten Merktzettel-Applikation und der Aufgabenliste als Gedankenstütze) und in den Fällen, wo ein stationärer PC oder auch ein Notebook zu groß oder zu unhandlich ist, aber trotzdem ein vollwertiger Computer benötigt wird. Vorrausgesetzt natürlich, man kann mit den gegebenen Einschränkungen leben (siehe Kapitel 2).

Die dafür zu entwerfenden palmbasierten CASE-Tools haben alle die gleiche Grundstruktur, können aber bei der Anzahl ihrer Ausstattungsmerkmale (Features) in drei Stufen variieren:

**Stufe 1:** Im einfachsten Fall nimmt die Applikation vom Nutzer Daten entgegen, wertet diese aus und zeigt das Resultat auf dem Bildschirm an.

---

<sup>28</sup><http://www.microtool.de>

Der Nutzer muss dann aber für eine weitere Verarbeitung und Speicherung des Ergebnisses selbst sorgen.

**Stufe 2:** Ein Programm der zweiten Stufe speichert die eingegebenen Daten und verwaltet sie projektbezogen. Optional könnte noch die Möglichkeit bestehen, die Ergebnisse mehrerer (Teil)Projekte miteinander zu vergleichen.

**Stufe 3:** Ein CASE-Tool der letzten Stufe besteht des Weiteren aus einem PC-Programm, welches die Palmdatenbank ausliest, aus den gewonnenen Informationen die Ergebnisse ermittelt und aus diesem Resultat und den Eingabedaten automatisch einen Bericht generiert. Eine Erweiterung dieser Stufe wäre es, die Eingabe der Informationen auch am PC zu ermöglichen und für den automatischen Datenabgleich mit dem Palm zu sorgen.

Es folgt eine Zusammenfassung und Konkretisierung der in den vorhergehenden Abschnitten gesammelten Ideen, zur Umsetzung von CASE-Tools für den Palm:

**P1:** In der Anfangsphase eines Softwareentwicklungsprozesses ist es wichtig, den zu erwartenden Aufwand (an Personal, Kosten und Zeit) abschätzen zu können. Eine verbreitete Möglichkeit ist die Function-Point Methode. Bei der Umsetzung sitzt der Zähler im Normalfall vor dem Dokument eines Softwareprojektes, welches nach Funktionseinheiten und Datenelementen hin untersucht werden soll. Der Palm ist sehr gut zur Unterstützung dieser Methode geeignet, da die manuelle Berechnung der Ergebnisse sehr zeitaufwendig ist und die Daten größtenteils aus einfachen Zahlen bestehen.

**P2:** Eine weitere Methode zur Aufwandschätzung ist das Constructive Cost Model (CoCoMo). Die Eingabe der Daten geschieht durch Auswahl von Prioritäten bezogen zu speziellen Schwerpunktthemen und durch Eingabe von Zahlenwerten. Die Ergebnisse werden auch hier durch ein komplexes Berechnungsmodell ermittelt, wodurch der Einsatz des Palms wiederum sinnvoll erscheint.

**P3:** Neben den beiden oben genannten Methoden zur Aufwandschätzung existieren in der Softwaretechnik für jede Phase des Softwarelebenszyklusses eine gewisse Anzahl von Softwaremaßen (siehe [Dum92] und

[DFKW96]). Eine Palmapplikation könnte die wichtigsten zusammenfassen und damit dem Entwicklungspersonal eines Softwaresystems eine metrikenbasierte Plattform, zur Untersuchung und Bewertung ihrer Arbeit bieten. Diese Plattform würde als *Erfahrungs-Datenbank* den Vergleich von Projekten, mit Hilfe der gesammelten Messdaten, ermöglichen, um damit als Resultat Aussagen und Abschätzungen über das aktuelle Produkt treffen zu können. Als Eingabedaten werden allerdings teilweise die Ergebnisse von Programmen benötigt, welche Dokumente und Quellcode automatisch nach speziellen Methoden analysieren (Software-Messtools). Hier wäre die Schaffung einer Schnittstelle, zum Import der benötigten Daten, erstrebenswert.

Der Nutzer sollte sich nach Start des Palmprogramms das gewünschte Maß aussuchen können, dann die Daten eingeben und zum Schluss die Ergebnisse präsentiert bekommen. Falls dem Nutzer die vorhanden Maße nicht bekannt sind, sollte sie das Programm zusätzlich den verschiedenen Lebenszyklusphasen zuordnen und möglicherweise kurz den Verwendungszweck mit angeben.

**P4:** Viele Firmen benutzen bei der Programmtestung vorgefertigte Fehlerprotokolle, in welche u.a. nur noch das getestete Softwaremodul, die Errormeldung oder Fehlerbeschreibung und der Weg zum Fehler eingetragen werden muss. Eine elektronische Umsetzung dieser Protokolle auf dem Palm, würde die Daten speichern und mehrere Möglichkeiten anbieten, um sie an den zuständigen Entwickler zu senden. Genutzt werden könnte dafür der Infrarotport, zur Kommunikation zweier Palms untereinander, das serielle Kabel, zur Kommunikation mit einem Desktop-Computer oder die Möglichkeit zum Anschluss eines Modems und der Nutzung des Internets, zur Weitergabe der Daten (z.B.: als Mail). Voraussetzung ist in den beiden ersten Kommunikationsformen eine Applikation auf der Zielplattform, die die Protokolle anzeigen und gegebenenfalls auch verwalten kann.

Durch den relativ großen Anteil an Schreibarbeit der hier durch den Nutzer zur Ausfüllung der Protokolle geleistet werden muss, bietet sich die Anschaffung einer externen Tastatur für den Palm an.

**P5:** Ein Programm zur Erstellung von Mängellisten sollte die gleichen Merkmale wie das Programm zur Erstellung von Fehlerprotokollen besitzen.

**P6:** Projektmanagemen-Pprogramme auf dem Palm sollten die in Abschnitt 3.2.1 auf Seite 27 beschriebenen Merkmale aufweisen können,

mit Berücksichtigung der Einschränkungen des Palms. Da es schon eine ganze Reihe sehr ausgereifter Produkte zu diesem Thema auf dem Markt gibt, ist eine weitere Betrachtung eigentlich nicht notwendig. Allerdings bietet es sich an, ein solches Tool als Dienst (Process-management service) bei der Entwicklung einer CASE-Plattform mit zu integrieren (siehe P9).

- P7:** Die Umsetzung eines Programms zur Zertifizierung eines Softwareentwicklungsprozesses nach der ISO-9000 Norm auf den Palm, muss alle 155 Fragen den Kategorien nach anzeigen können und dem Nutzer die Möglichkeit bieten, zwischen einer positiven oder einer negativen Bewertung zu wählen. Bei der Auswertung sollte die Prozentzahl der positiv beantworteten Fragen einmal gesamt und einmal in Bezug auf die jeweilige Kategorie hin, angezeigt werden.
- P8:** Die palmbasierte Unterstützung des Capability Maturity Modells (CMM) ist analog zur ISO-9000 Methode aufgebaut, mit dem Unterschied, dass die Fragen zusätzlich mit den Levels zu kennzeichnen sind, zu deren Erreichung eine positive Beantwortung notwendig ist. Außerdem sollte bei der Gesamtbewertung die resultierende Stufe des Entwicklungsprozesses mit angegeben werden.
- P9:** Eine gemeinsame Plattform, für die hier beschriebenen CASE-Tools, müsste aus folgenden Elementen bestehen:
- 1. Einer allgemeinen Datenbank,** mit Informationen über alle Dokumente, Quellkodateien und Ressourcen der verschiedenen Projekte.
  - 2. Einer Funktionsbibliothek,** mit Funktionen zum Zugriff der CASE-Tools und Dienste auf die Datenbank.
  - 3. Einem Datenbank-Frontend,** zur Erzeugung, Bearbeitung und Entfernung von Einträgen.
  - 4. Einem Prozess-Management-Dienst,** zur Projektverwaltung.

Außerdem ist es notwendig, dass die einzelnen CASE-Tools den Aufbau ihrer eigenen Tool-Datenbanken offen legen, damit andere CASE-Tools auf die darin enthaltenen Informationen zugreifen können (z.B. finden die mit einem Function-Point Zähltool ermittelten Ergebnisse in der Version 2.0 der CoCoMo-Methode Verwendung). Die Sicherheit der

Datenbestände der Projekt- und Tooldatenbanken erfordert es, dass alle Zugriffe ausschließlich über die Funktionsbibliothek erfolgen, die mit Mechanismen zur Authentisierung von Tool oder Dienst und zugehöriger Datenbank ausgestattet sein sollte.

Damit das ganze nicht zum Selbstzweck verkommt, müssen auch Programme auf dem PC existieren, die die Sicherung der Datenbanken übernehmen und die darin enthaltenen Informationen zur Weiterverarbeitung z.B. durch Textverarbeitungsprogramme oder Tabellenkalkulationen aufbereiten (Stufe 3).

## 4 CASE-Tools für den Palm

In diesem Kapitel werden einige der im Abschnitt 3.4 aufgezählten CASE-Tools kurz beschrieben. Als Modellierungssprache dient die Unified Modeling Language (UML).

### 4.1 Eine metrikbasierten Erfahrungs-Datenbank für den Palm (P3)

Um die Funktionsweise und den Zweck einer metrikbasierten Erfahrungs-Datenbank für Softwareprojekte erklären zu können, müssen erst einmal die Begriffe Software-Metrik und Software-Maß definiert werden:

„Eine **Software-Metrik** (*software metric*) ist gemäß der **Maßtheorie** (*measure theory*) eine **Abstandsfunktion** (*distance*), die Attribute von Software-Komponenten Zahlen (-bereiche) zuordnet.“ [Dum00, Seite 158]

„Ein **Software-Maß** (*software measure*) ist gemäß der **Messtheorie** (*measurement theory*) eine mit einer **Maßeinheit**<sup>29</sup> (*unit*) versehene **Skala** (*scale*), die in dieser Form ein Software-Attribut bewertet bzw. messbar macht.“ [Dum00, Seite 159]

Eine metrikbasierte Erfahrungs-Datenbank dient zum Sammeln und Auswerten von Messergebnissen, welche im Laufe des Lebenszyklusses eines Softwareproduktes in jeder Phase entstehen können. Anhand empirischer Untersuchungen können diese Daten mit Hilfe von Software-Metriken genutzt werden, um Aufwandschätzungen vorzunehmen, um Qualitätsaussagen zu treffen oder um Software-Komponenten und Entwicklungsprozesse zu bewerten. Dabei existieren laut [She81, Seite 102–120] folgende Messgegenstände:

- Programmcode,
- Programmierungsmethoden,
- Programmierungsaspekte und
- Programmierungsmanagement.

---

<sup>29</sup>Gegebenfalls wird auch nominales oder ordinales "Messen" zugelassen. Dann entfällt die Maßeinheit.

Hier ein paar Beispiele für Software-Maße und -Metriken:

**Lines of Code (LOC)** Beim LOC-Maß wird die Anzahl der Quellcodezeilen gemessen, um damit den Umfang eines Programms zu ermitteln. Unterschieden wird dabei u.a. in abarbeitbare Zeilen, Datendefinitionen, Steuerungsanweisungen und Kommentarzeilen.

**Object-Points** Die Object-Points dienen wie die Function-Points zur Aufwandschätzung von Softwareprojekten. Im Unterschied dazu, wird eine Produktbeschreibung nicht auf ihren "funktionellen Gehalt" sondern auf die Anzahl von "screens", "reports" und "third-generation language components" hin untersucht (siehe auch [Pff99, ab Seite 106]). Dadurch kann diese Methode in einem früheren Zeitpunkt im Lebenszyklus eines Software-Projektes eingesetzt werden.

**Halstead-Metrik** Die Halstead-Metrik von M. H. Halstead entstand in den 70-er Jahren und dient u.a. zur Ermittlung der Programmlänge, des Programmumfanges, des Programmieraufwandes und des Schwierigkeitsgrades. Dazu müssen folgende Ausgangsgrößen gemessen werden:

- $\eta_1$  = Anzahl unterschiedlicher Operatoren,
- $\eta_2$  = Anzahl unterschiedlicher Operanden,
- $N_1$  = Gesamtzahl der Operatoren im Programm,
- $N_2$  = Gesamtzahl der Operanden im Programm.

Mehr zur Halstead-Metrik gibt es in [Dum92, ab Seite 91].

**McCabe-Metrik** Die McCabe-Metrik (Beschreibung in [Dum92, ab Seite 102]) dient der Ermittlung der Programmkomplexität durch die Berechnung der zyklomatischen Zahl eines dem Programm zugehörigem Programmflussgraphen mit den Ausgangsgrößen: Knotenanzahl ( $e$ ), Kantenanzahl ( $n$ ) und der Anzahl der zusammenhängenden Komponenten ( $p$ ). Die Berechnung erfolgt mit der Formel:  $v(G) = e - n + 2p$ .

Die Metriken-Datenbank sollte folgenden Anforderungen genügen:

- Projektweise Abspeicherung aller eingegebenen Messdaten,
- Unterteilung der Projekte in Software-Module oder -Komponenten,

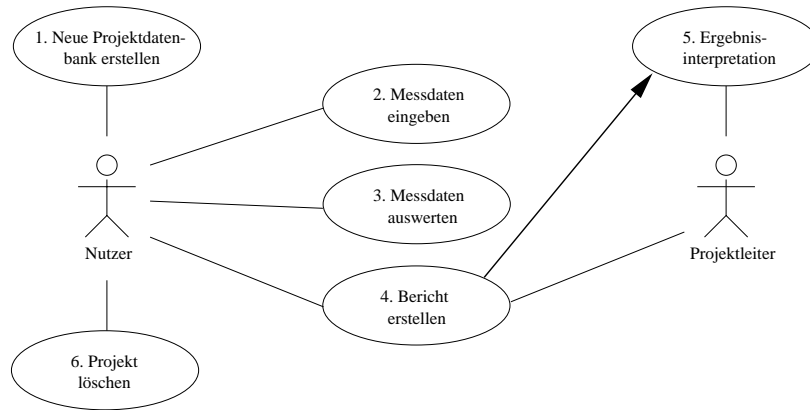


Abbildung 10: Anwendungsfälle der Metrik-Datenbank

- Zuteilung der Metriken zu den sechs Metrik-Gruppen (Metriken für die Anforderungsanalyse, Spezifikations-Metriken, Design-Metriken, Code-Metriken, Test-Metriken und Wartungsmetriken),
- Berechnung und Präsentation von Ergebnissen,
- Grafische Darstellung von Messdaten und -ergebnissen (soweit möglich),
- Modularer Aufbau des Programms, zur einfachen Erweiterung mit neuen Maßen und Metriken, auch durch den Nutzer,
- Schaffung von Möglichkeiten, zum Austausch von Projektdatenbanken.
- Schaffung von Möglichkeiten, zum Vergleich von Messdaten zweier verschiedener Projekte,
- Wenn vom Nutzer gewünscht, Anzeige einer kurzen Benennung oder Beschreibung der Implementierten Maße und Metriken,
- Schaffung einer Desktopapplikation, zur Sicherung der Daten und zur Berichtgenerierung.

#### 4.1.1 Szenariobeschreibung

Abbildung 10 zeigt die Akteure und Anwendungsfälle der Metrik-Datenbank. Auf der einen Seite steht der Nutzer (z.B.: Programmierer, Software-Ingenieur, Qualitätssicherer usw.), welcher die Projektdatenbanken anlegt,

verwaltet und löscht, die Messdaten eingibt, erste Auswertungen vornehmen kann und mit Hilfe einer Desktop-Applikation, aus den Messdaten und Ergebnissen, einen Bericht generiert. Ein weiterer Akteur (hier als Beispiel der Projektleiter) kann anhand des Berichtes und empirischer Erfahrungen (z.B. durch Vergleich der Ergebnisse mit denen älterer Projekte) eine Ergebnisinterpretation vornehmen und die so gewonnenen Erkenntnisse in den Entwicklungsprozess mit einbeziehen. Beispiele dafür sind Abschätzungen der zu erwartenden Kosten, Aussagen über die Qualität der Software und Bewertung der Produktivität von Entwicklungsteams.

Startet der Nutzer das Palm-Programm, sollte er als erstes eine Liste mit den vorhandenen Projektdatenbanken angezeigt bekommen und die Möglichkeiten haben, eine neue Projektdatenbank anzulegen, Messdaten einzugeben, Messdaten auswerten zu lassen oder ein Projekt zu löschen. Wurde die erste Möglichkeit gewählt (Anwendungsfall 1), sollte das Programm vom Nutzer einen Namen für des Projekt verlangen sowie möglicherweise den Start- und wenn bekannt, den Endzeitpunkt und daraufhin die Datenbank generieren. Da ein Software-Projekt in den meisten Fällen aus mehreren Komponenten oder Modulen besteht, sollten zur Erhöhung der Übersichtlichkeit, diese noch mit angegeben werden, um später die einzugebenen Messdaten diesen Komponenten zuordnen zu können.

Hat der Nutzer alle notwendigen Angaben gemacht, kann er mit der Ein-

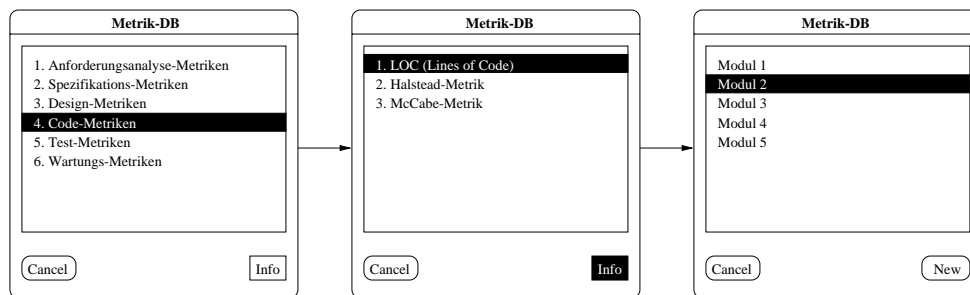


Abbildung 11: Einstellungen zur Eingabe der Messdaten

gabe der Messdaten beginnen (Anwendungsfall 2). Dazu muss aber noch die Metrik-Gruppe (siehe Anforderungen), die Metrik, zu der die Messdaten eingegeben werden sollen und der Name der analysierten Software-Komponente mit angegeben werden (Beispielbildschirme siehe Abbildung 11). Dem Nutzer sollten bei diesen drei Eingaben, grundsätzlich alle möglichen Alternativen, beispielsweise in Listenform angezeigt werden, so dass er sich eine davon, durch Markierung, aussuchen kann. Zusätzlich sollte die Option existieren,

Einträge für neue Software-Komponenten zu erstellen, falls dies nötig ist. Nachdem alle notwendige Einstellungen getroffen wurden, braucht der Nutzer nur noch die Messergebnisse in die vorbereiteten Bildschirme eintragen und die Daten durch das Programm in der Datenbank abspeichern lassen. Der dritte Anwendungsfall betrifft die Auswertung von Projekten durch Anzeige der Messdaten, die Berechnung und Anzeige der Ergebnisse und der Vergleich von Daten und Ergebnissen mit denen anderer Projekte. Dazu sind folgende Möglichkeiten denkbar:

- Unterscheidung, ob Messdaten und Ergebnisse für die einzelnen Software-Komponenten oder summiert für das Projekt angezeigt werden sollen.
- Sollen nur Messdaten, nur Ergebnisse oder beides angezeigt werden.
- Soll nur eine einzelne Metrik oder eine gesamte Metriken-Gruppe betrachtet werden.
- Soweit möglich, Darstellung der Messdaten und Ergebnisse in Linien-, Balken-, oder Kiviatt-Diagrammen.
- Tabellarische Darstellung von Messdaten und Ergebnissen von verschiedenen Projekten zur Gegenüberstellung.
- Sind für Metriken Grenzwerte bekannt (z.B.: beim Verhältnis Codezeilen zu Kommentarzeilen), die nicht über- oder unterschritten werden sollten, ist ein Verstoß sofort anzuzeigen (z.B. könnte bei farbfähigen Palms der entsprechende Wert Rot dargestellt werden).
- Auswahl von einzelnen Metriken und Messdaten zur Darstellung.

Da es sich hier im weitesten Sinne um eine Datenbankanwendung handelt, sollte das Programm bei der Darstellung von Messdaten und Ergebnissen möglichst flexibel auf die Wünsche des Nutzers eingehen, damit dieser auch wirklich die Informationen bekommt die er braucht, um daraus die erhofften Rückschlüsse ziehen zu können.

Für die Sicherung der Projektdatenbanken auf einem Desktop-Computer und die abschließende Berichtgenerierung (Anwendungsfall 5) ist ein PC-Programm zuständig. Hier sollte der Nutzer die gleichen Einstellungen zur Auswertung der Datenbanken tätigen können wie auf dem Palm (siehe Aufzählung weiter oben). Als Ausgabeformat für die Berichte bieten

sich unformatierte Textdateien im ASCII-Format, formatierte Textdokumente im RTF-Format und Tabellen für die gängigen Tabellenkalkulations-Programme an. Die Generierung der Berichte sollte bis auf das Einstellen von Eingangsparametern durch den Nutzer, vollständig automatisch ablaufen. Daraufhin kann die Weiterverarbeitung, mit entsprechenden Programmen, manuell erfolgen.

Soll eine der Projektdatenbanken vom Palm entfernt werden (Anwendungsfall 6), z.B. um Speicherplatz zu sparen, kann dies mit Hilfe der Projektliste im Hauptbildschirm und mit einem zugehörigen Button geschehen.

#### 4.1.2 Die Datenbank der Palm-Applikation (MetrikDB)

Palmdatenbanken sind recordbasiert aufgebaut (siehe Abschnitt 2.5). Daher wäre es am einfachsten, die Projekte und Module einfach hintereinander als Records inklusive der Messergebnisse mit folgender Struktur in die Datenbank zu schreiben: *Projektrecord 1, Modulrecord 1, Modulrecord 2, ..., Modulrecord n, Projektrecord 2, Modulrecord 1, ..., Modulrecord m, Projektrecord 3 usw.* Im Projektrecord würde die ID-Nummer, der Name und weiterführende Informationen sowie die Anzahl der Module stehen. Der Modulrecord würde ID-Nummer, Name, weiterführende Informationen und die Messdaten enthalten. Durch die Abspeicherung der Anzahl der Module im Projektrecord, weiß das Programm immer, welche Module zum Projekt gehören und wo das nächste Projekt zu erreichen ist. Durch die Annahme, dass die Anzahl der Projekte in einer palmbasierten Erfahrungsdatenbank eher gering bleibt (maximal vier bis acht Projekte), bewegen sich damit auch die Zugriffszeiten auf einzelne Records auf einem vernachlässigbaren Niveau.

Allerdings hat diese Methode zur Verwaltung der Datenbestände einige gravierende Nachteile. Erstens ist es bei vordefinierten Records nicht mehr möglich neue Messdatenarten und Metrikarten durch den Nutzer einzuführen, zweitens ist die Kapazitätsgrenze einer Palmdatenbank von 32 kB bei größeren Projekten schnell erreicht und drittens ist es sehr aufwendig, Projektdatenbanken zwischen zwei Palm-Computern auszutauschen. Eine bessere, wenn auch etwas kompliziertere Lösung ist es, für jedes Projekt eine extra Palmdatenbank anzulegen. Damit sind allerdings noch nicht alle Probleme gelöst. Zusätzlich braucht das Programm noch eine Datenbank, wo die Zugriffsnamen aller Projektdatenbanken abgelegt werden und eine Datenbank, wo die Definition aller vom Nutzer hinzugefügten Metriken gespeichert wird. Abbildung 12 zeigt das ERM-Modell der Erfahrungs-Datenbank.

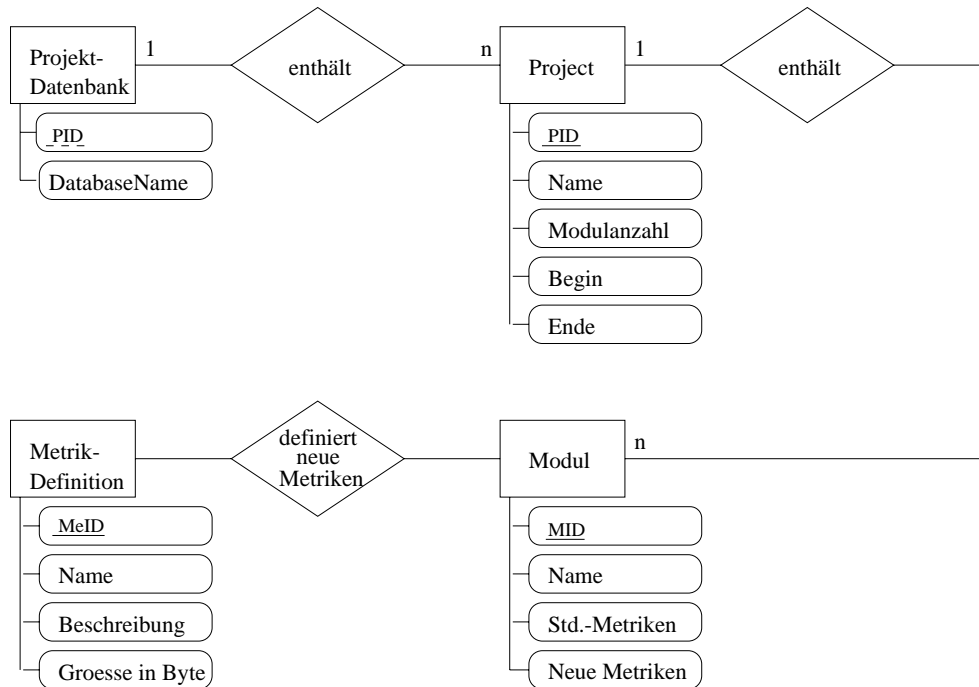


Abbildung 12: ERM-Modell der MetrikDB

Das *Std.-Metriken* Attribut (siehe Modul Entity) steht für alle im Programm standardmäßig enthaltenen Messdaten (es handelt sich also strenggenommen um mehrere Attribute) und das *Neue Metriken* Attribut enthält die Messdaten und Ergebnisse aller vom Nutzer definierten Metriken (siehe Metrik-Definitions Entity).

Die Palmdatenbanken von MetrikDB könnten also wie folgt aufgebaut werden:

1. Datenbank mit Projektnamen
Projektdatenbankname 1
Projektdatenbankname 2
Projektdatenbankname 3
...
Projektdatenbankname n

<b>2. Datenbank mit Metrikdefinitionen</b>
Metrikdefinition 1
Metrikdefinition 2
Metrikdefinition 3
...
Metrikdefinition n
<b>3.-n. Projektdatenbanken</b>
Projektrecord
Modulrecord 1
Modulrecord 2
Modulrecord 3
...
Modulrecord n

Es existiert also eine Datenbank mit den Namen der Projektdatenbanken, eine Datenbank mit den Definitionen der hinzugefügten Metriken und eine beliebige Anzahl an Projektdatenbanken.

Die Aufnahme des Projektrecords in die Projektdatenbank scheint auf den ersten Blick redundant zu sein. Die dort enthaltenen Informationen könnten auch in der Datenbank mit den Namen der Projekte abgespeichert werden. Allerdings wird damit ein bestimmtes Ziel verfolgt. So können durch ein einfaches Übersenden der Projektdatenbank über Infrarot auf ein anderen Palm, die Daten in die dortige Erfahrungs-Datenbank mit übernommen werden. Beim Ziel-Palm muss nur noch der Name der Datenbank angegeben werden und wenn nötig, die Definition der hinzugefügten Metrikentypen. Letzteres kann allerdings durch eine Importfunktion automatisiert werden.

#### 4.1.3 Die Zustände der Metrik-Datenbank

Anhand der Abbildung 13 sollen die Zustände der Palm-Applikation aus Nutzersicht, mit den verschiedenen Bildschirmen des Programms dargestellt werden. Dabei orientiert sich das Diagramm an der Szenariobeschreibung und den Anwendungsfällen aus Abschnitt 4.1.1. Die Pfeile stellen die Wege dar, auf denen der Nutzer von einem Bildschirm oder von einer Bildschirmgruppe zum nächsten Bildschirm oder zur nächsten Bildschirmgruppe gelangt. Die Anzahl der Bildschirme ist abhängig vom Design und vom Aufbau der je-

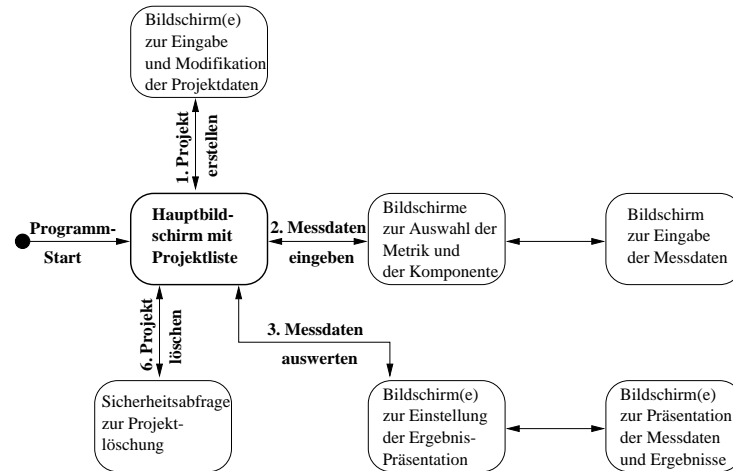


Abbildung 13: Die Zustände der Palm-Applikation (MetrikDB)

weiligen Bildschirminhalte sowie von der Anzahl der dargestellten oder vom Nutzer einzugebenden Informationen.

#### 4.1.4 Die Komponenten und Klassen der Metrik-Datenbank

Da Palm-Applikationen hauptsächlich in "C" geschrieben werden, wird hier zur Darstellung des Aufbaus das Komponenten-Diagramm verwendet. Im Gegensatz dazu steht für die Desktop-Applikation das Klassendiagramm zur Verfügung, da dort die am meisten verwendeten Programmiersprachen objektorientiert sind (z.B.: "C++", "Visuall Basic" und "Java").

Die Palm-Applikation besteht aus folgenden Komponenten (siehe Abbildung 14):

**Datenbank** Die Datenbank enthält alle vom Nutzer eingegebenen Informationen. Die Palm-Applikation MetrikDB kann über Funktionen des PalmOS darauf zugreifen.

**Datenbankmanager** Der Datenbankmanager besteht aus Funktionen zum Zugriff auf die Datenbank der Palm-Applikation. Dazu gehört das Anlegen einer Datenbank, das Öffnen der Datenbank, das Schreiben von Informationen, das Herausholen von Informationen und das Löschen von Datenbankeinträgen. Alle anderen Komponenten können die Funktionen des Datenbankmanagers nutzen, um auf die Datenbank zuzugreifen.

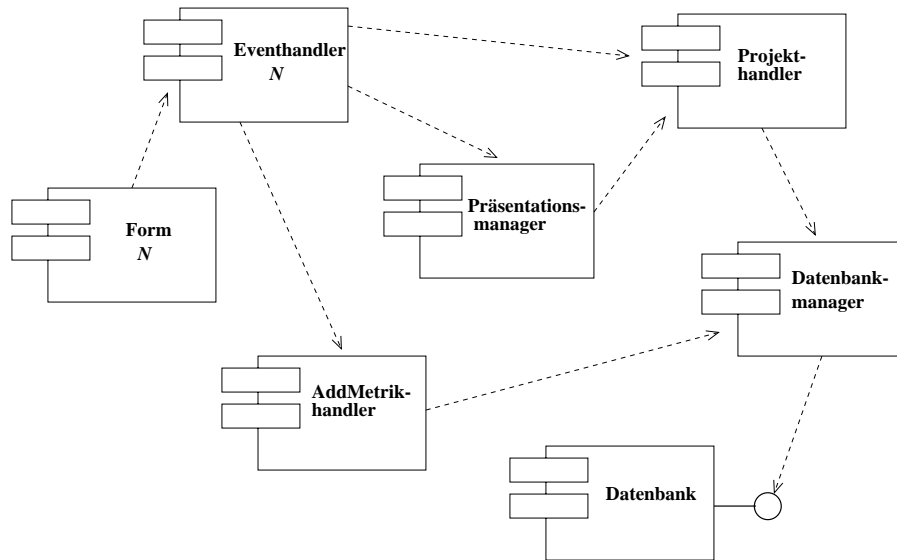


Abbildung 14: Die Komponenten der Palm-Applikation

**Form** Eine Form ist die Definition eines Programm-Bildschirms mit all seinen Inhalten, wie Buttons, Listen, Felder usw. Die Metrik-Datenbank enthält eine größere Anzahl solcher Forms, die alle in einem Ressourcen-Definitions-File beschrieben werden.

**Eventhandler** Palm-Programme bestehen gewöhnlicherweise aus mehreren Eventhandlern. Einen für den Start und einen für das Ende der Applikation und jeweils einen für jede Form. Der Eventhandler hat dabei die Aufgabe, angemessen auf die zugehörigen Ereignisse in der entsprechenden Form zu reagieren.

**Projekthandler** Der Projekthandler verwaltet die einzelnen Projekte der Applikation. Dazu gehört es, durch Nutzung des Datenbankmanagers, neue Projekte anzulegen, die vom Nutzer eingegebenen Messdaten dem Projekt zugehörig abzuspeichern, den Zugriff auf die Daten für andere Komponenten zu ermöglichen, Daten zu manipulieren und wenn nötig, Projekte zu entfernen.

**Präsentationsmanager** Der Präsentationsmanager dient zur Berechnung und Darstellung der Metriken. Desweiteren hat er die Aufgabe, die vom Nutzer getätigten Einstellung zur Aufbereitung der Ergebnis-Präsentation umzusetzen. Zum Zugriff auf die Datenbestände der Projekte

werden die Dienste des Projektmanagers benötigt und für die Informationen über die vom Nutzer gewünschten Einstellungen, die Funktionen des zur entsprechenden Form gehörenden Formhandler.

**AddMetrikhandler** Die Aufgabe des AddMetrikhandlers ist es, neue Metriken vom Nutzer entgegenzunehmen und das Format für die Weiterverwendung entsprechend aufzubereiten. Über den Datenbank-Manager werden die neuen Metriken in der Metrik-Definitions-Datenbank abgelegt.

Die Desktop-Applikation zur Auswertung der Datenbanken und zur Berichterstattung hat die gleichen Aufgaben und den gleichen Aufbau wie das entsprechende Programm des Function-Point Zähltools (siehe Abschnitt 5.7). Daher kommt das dort verwendete Klassendiagramm auch hier zum Einsatz. Unterschiede gibt es nur in dem Aufbau der Datenbank und den verwendeten Datenstrukturen sowie in den Auswahlmöglichkeiten zum Aussehen und Inhalt der zu generierenden Berichte.

#### 4.1.5 Fazit und Ausblick

Der Vorteil einer mobilen Metrik-Datenbank, in der alle auf Messungen beruhenden Erfahrungen eines Projektes gespeichert und jederzeit abrufbar sind, liegen auf der Hand. Der Nutzer ist unabhängig von Desktop-Computern und hat die benötigten Information als Gedankenstütze immer verfügbar, wenn er beispielsweise Aussagen mit Zahlen untermauern oder bei Präsentationen konkrete Fragen nach Ergebnissen, Verbesserungen oder Abschätzungen beantworten will. Allerdings sind durch die technischen Grundlagen des Palms einige Einschränkungen gegeben. Bedingt durch die geringe Bildschirmgröße und fehlende Anschlussmöglichkeiten an externe Sichtgeräte, verbietet sich der Palm zum Einsatz als Präsentationswerkzeug. Des weiteren sollte die Größe und die Anzahl der Projekte nicht allzu umfangreich sein, da Palm-Computer nur über einen eingeschränkten Speicherumfang und über relativ geringe Rechenleistungen verfügen. Mit kleineren "überschaubaren" Projekten kann der Palm dagegen sehr gut umgehen. Eine Verbesserung der Situation kann durch die Aufhebung der "Palmbasiertheit" erreicht werden. Damit ist gemeint, dass die eigentliche Erfahrungsdatenbank auf ein Desktop-Computer verlegt und der Palm dagegen als Frontend genutzt wird. Der Desktop-Computer kommt dann zur Verwaltung und Speicherung der Datenbank zum Einsatz und auf den Palm werden nur noch die Projekte und

Daten per Synchronisation übermittelt, die der Nutzer benötigt oder bearbeiten will. Wurden auf dem Palm Daten bearbeitet oder hinzugefügt, können diese per Hotsync-Vorgang in die Datenbank überspielt werden. Durch entsprechende Vorkehrungen zur Konsistenzwahrung des Datenbestandes, ist sogar eine verteilte Nutzung der Erfahrungsdatenbank, z.B. durch ein komplettes Entwicklungsteam, bei dem jedes Mitglied über einen Palm verfügt, denkbar. Dabei muss nicht einmal ein direkter Zugang zu dem Desktop-Computer bestehen, da die Daten auch durch Nutzung eines Modems, über das Internet, ausgetauscht werden können.

## **4.2 Ein palmbasiertes Formular zur Erstellung und Anzeige von Fehler-Protokollen (P4)**

Bei vielen Softwareprojekten nutzen Programmtester vordefinierte Fehlerformulare mit speziellen Eingabefeldern zur Beschreibung von Programmfehlern. Im einfachsten Fall bestehen diese aus vorgedruckten Papierbögen, welche durch den Tester ausgefüllt und dann manuell oder per FAX an den zuständigen Programmierer weitergegeben werden. Eine andere Möglichkeit ist die Nutzung eines einfachen Texteditors oder einer Textverarbeitung auf einem Desktop-Computer. Die auf diese Weise entstandenen Fehlerprotokolle kann der Programmtester per EMail verschicken oder sie dem Programmierer, beispielsweise auf einer gemeinsam nutzbaren Festplatte im Firmen-Netzwerk, verfügbar machen.

Die Vorteile bei der elektronischen Verarbeitung der Fehler-Reports liegen in der einfacheren Verwaltung von Dateien oder E-mails im Gegensatz zu einer losen Blattsammlung, in der Einsparung an Kosten (z.B.: für den Druck der Formulare) und in der Vermeidung von Missverständnissen, ausgelöst durch Schwierigkeiten bei der Entzifferung der handgeschriebenen Berichte.

Der Einsatz des Palms, zur Erstellung und Verwaltung von computergestützten Fehler-Protokollen, macht u.a. in den Fällen Sinn, wo der Programmtester bei der Überprüfung eines Softwaremodul keinen Zugriff auf einen Desktop-Computer mit einer Applikation hat, die es ihm ermöglicht diese Fehler-Protokolle zu erstellen. Ein Beispiel dafür ist die Testung von Software auf Maschinen und Anlagen und auf Embedded Systems.

Bei der Weitergabe der Fehlerprotokolle kann eine dementsprechenden Applikation alle dem Palm zur Verfügung stehenden Möglichkeiten zu Kommunikation nutzen:

**Infrarot:** Die erste Möglichkeit geht davon aus, dass dem Programmierer auch ein Palm-Computer zur Verfügung steht. In diesem Fall können die Daten vom Palm des Testers, direkt über den Infrarotport, an den Palm des Programmierers gesendet werden.

**Seriell:** Der Palm kann seriell (Kabel oder Infrarot) mit dem Desktop-Computer des Programmierers verbunden werden und auf diese Weise die Protokolle übertragen. Dazu muss auf dem Zielrechner allerdings eine Applikation existieren, die die Protokolle entgegennimmt und verwaltet.

**EMail:** Es existieren zwei Möglichkeiten die Protokolle per EMail an den Programmierer zu senden. Entweder wird ein Modem direkt an den Palm angeschlossen oder die Mail wird beim Hotsync-Vorgang an den PC des Testers übertragen und dann, von dort aus, weitergeleitet. Bei der Nutzung dieser Methode kann auf eine spezielle Applikation auf dem Zielrechner verzichtet werden.

**Hotsync:** In diesem Fall übernimmt eine Desktop-Applikation die Weitergabe der Fehler-Protokolle. Beim Hotsync-Vorgang wird die Palm-Datenbank mit den ausgefüllten Formularen auf dem PC gesichert, die Daten werden extrahiert und können dann auf verschiedenen Wegen den Programmierer erreichen. Beispielsweise durch Kopieren der Daten auf ein gemeinsam genutztes Netzlaufwerk oder wieder durch Verschieben per EMail.

Die ersten beiden Kommunikationsarten kommen hauptsächlich zum Einsatz, wenn der Tester und der Programmierer in direktem Kontakt miteinander stehen und die anderen beiden, wenn der Kontakt zwischen Tester und Programmierer auf einem gewissen räumlichen Abstand beruht.

Ein palmbasiertes Tool zur Erstellung, Verwaltung und Verteilung von Fehler-Protokollen sollte folgenden Anforderungen genügen:

- Alle eingegebenen Daten sollten in einer Datenbank abgespeichert werden,
- Es sollten Bildschirme zu Anzeige von bestehenden Fehler-Protokollen existieren,
- Protokolle sind mit dem Status ihrer Bearbeitung kennzeichenbar sein (Fehler beseitigt? ja/nein),

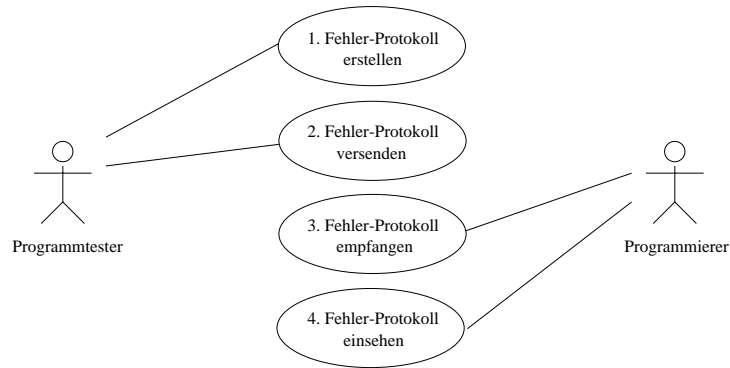


Abbildung 15: Anwendungsfälle des Fehler-Protokoll Editors

- Alle oben genannten Kommunikationsformen sind zu unterstützen,
- Eine Desktop-Applikation sollte Fehlerprotokolle vom Palm entgegennehmen, verwalten und verteilen können,
- Folgende Informationen sollten in das Protokollformular eingegeben werden können:
  - Name des Testers,
  - Name des Projektes,
  - Name des Modules,
  - Vom Programm ausgegebene Fehlermeldung (meist eine Zahlen/Ziffern-Kombination),
  - Beschreibung der Auswirkungen des Fehlers (z.B.: Absturz, Fehlerberechnung, falsche Darstellung usw.)
  - Beschreibung des Weges zum Fehler,
  - Grad der Ernsthaftigkeit des Fehlers, beispielsweise auf einer Skala von eins bis fünf (eins: harmlos, 5: gravierend),
  - Bearbeitungsstatus.

#### 4.2.1 Szenariobeschreibung

Es gibt zwei verschiedene Möglichkeiten für den Aufbau dieses Werkzeuges. Entweder wird für jeden Nutzer (siehe Anwendungsfalldiagramm in Abbildung 15) eine eigene autonome Palmapplikation erstellt oder eine Applikation übernimmt alle Aufgaben. Auf jeden Fall kommt noch ein PC-Programm

zur Entgegennahme und Anzeige der Fehlerprotokolle auf einem Desktop-Computer hinzu. Der Einfachheit halber, wird bei diesem Entwurf von einer gemeinsamen Palm-Applikation für Tester und Programmierer ausgegangen. So kann im Fall, dass eine Person beide Programmteile nutzen möchte, Speicherplatz auf dem Palm gespart werden, da Tester und Programmierer, einige Funktionen gleichsam nutzen (z.B.: Routinen zur Verwaltung der Protokolle und zur Kommunikation).

Der Startbildschirm sollte in Abhängigkeit vom Nutzer variieren. Beim Te-

Abbildung 16: Bildschirme zur Erstellung eines neuen Protokolls

ster ist es am sinnvollsten, als erstes ein leeres Formular, zur Eingabe des gefundenen Programmfehlers anzuzeigen (siehe Abbildung 16). Nach Fertigstellung kann dieser dann das Protokoll, mit einem Button, in der Datenbank abspeichern und mit dem nächsten Fehler fortfahren. Hat er den Wunsch die Protokolle weiterzuleiten, sind entsprechende Bildschirme notwendig, die die verschiedenen Kommunikationsmöglichkeiten anbieten.

Dem Programmierer dagegen, sollte als aller erstes eine Übersicht mit allen in der Datenbank vorhandenen Protokollen angezeigt werden, inklusiver der Informationen, ob die Fehlermeldung neu ist oder schon bearbeitet wurde. Hat er eine neue erhalten, könnte der Palm ihn mit einer Nachricht darauf aufmerksam machen. Mit Hilfe der Liste und eines entsprechenden Buttons besteht dann die Möglichkeit, sich Fehlermeldungen anzeigen zu lassen, als Erledigt zu markieren oder aus der Datenbank zu löschen.

Die Desktop-Applikation übernimmt folgende Aufgaben:

- Direkte Kontaktaufnahme zum Palm, über serielle Verbindung, zur Annahme der Protokolle,
- Entgegennahme der Palmdatenbank bei einem Hotsync-Vorgang und Extrahierung der Protokolle,

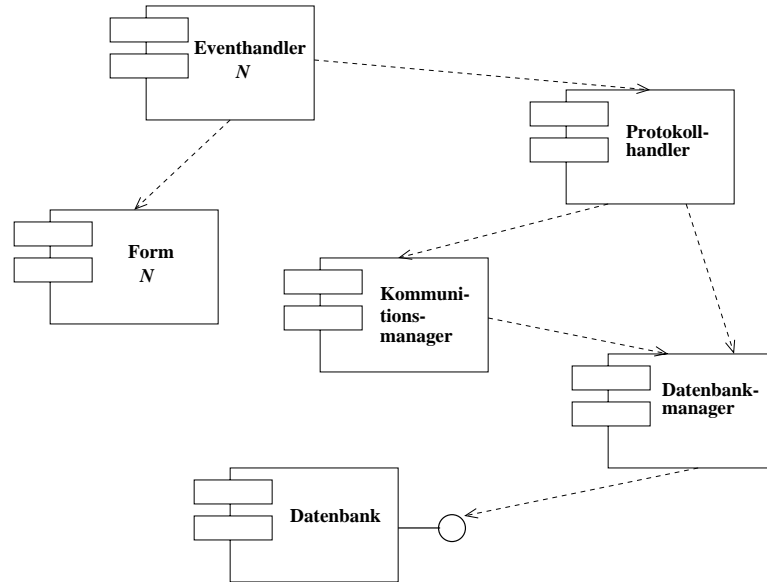


Abbildung 17: Die Komponenten der Palmapplikation

- Konvertierung der Protokolle in der Palmdatenbank, beispielsweise in eine ASCII- oder Textverarbeitungsdatei,
- Versenden der Protokolle per EMail oder durch Kopieren in ein entsprechendes Verzeichnis,
- Anzeige und Verwaltung der Protokolle.

#### 4.2.2 Die Komponenten der Palmapplikation

In Abbildung 17 sind die Beziehungen zwischen den Komponenten der Palm-Applikation grafisch dargestellt. Die *Eventhandler* verarbeiten Ereignisse in den zugehörigen *Forms*. Werden bei einem Ereignis Protokolldaten eingegeben, modifiziert oder angefordert, so beauftragt der Eventhandler den *Protokollmanager*. Dieser kann die Funktionen des *Datenbankmanagers* nutzen, wenn das entsprechende Protokoll in der *Datenbank* vorliegt oder dort gespeichert werden soll oder er nutzt die Funktionen des *Kommunikationsmanagers*, um Protokolle zu verschicken oder zu empfangen.

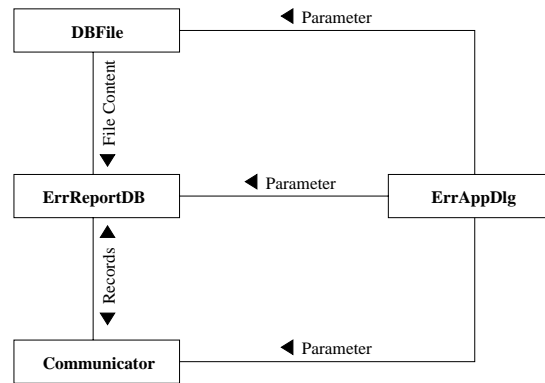


Abbildung 18: Die Klassen der Fehler-Protokoll Desktop-Applikation

### 4.2.3 Die Klassen der Fehler-Protokoll Desktop-Applikation

Abbildung 18 zeigt einen groben Überblick über die möglichen Klassen der Desktop-Applikation des Fehler-Protokoll Editors. *ErrAppDlg* kümmert sich um das Userinterface der Applikation und verwaltet und bedient die Eingaben des Nutzers. Die Klasse *DBFile* enthält Methoden zum Extrahieren der Fehler-Protokolle aus der auf dem PC gesicherten Palmdatenbank, welche von der Klasse *ErrReportDB* verwaltet werden. Der *Communicator* dient zum Empfangen von Fehlermeldungen (z.B. bei der Direktverbindung Palm-PC) oder zu ihrer Versendung (z.B.: per EMail). Dazu nutzt er die Dienste von *ErrReportDB*.

### 4.2.4 Kommunikation

Die Übertragung der Fehler-Protokolle vom Palm des Programmtesters per Infrarot auf den Palm des Programmierers wird mit einer Anfrage an den Kommunikationsmanager des Fehler-Protokoll Editors gestartet (siehe Abbildung 19). Daraufhin versucht dieser, unter Ausnutzung der Infrarot-Library des PalmOS, mit einem anderen Palm in Reichweite des Infrarotsensors Kontakt aufzunehmen. Wird der Verbindungswunsch bestätigt, fängt der Palm des Testers an die Daten zu versenden. Gleichzeitig erscheint auf dem Display des Quell- und des Zielpalms eine dementsprechende Nachricht. Nach der erfolgreichen Übertragung der Fehler-Protokolle, wird den Nutzern dieses mitgeteilt und der Programmierer gefragt, ob er die Daten annehmen möchte. Nach dessen Bestätigung werden die Fehlerprotokolle der Datenbank

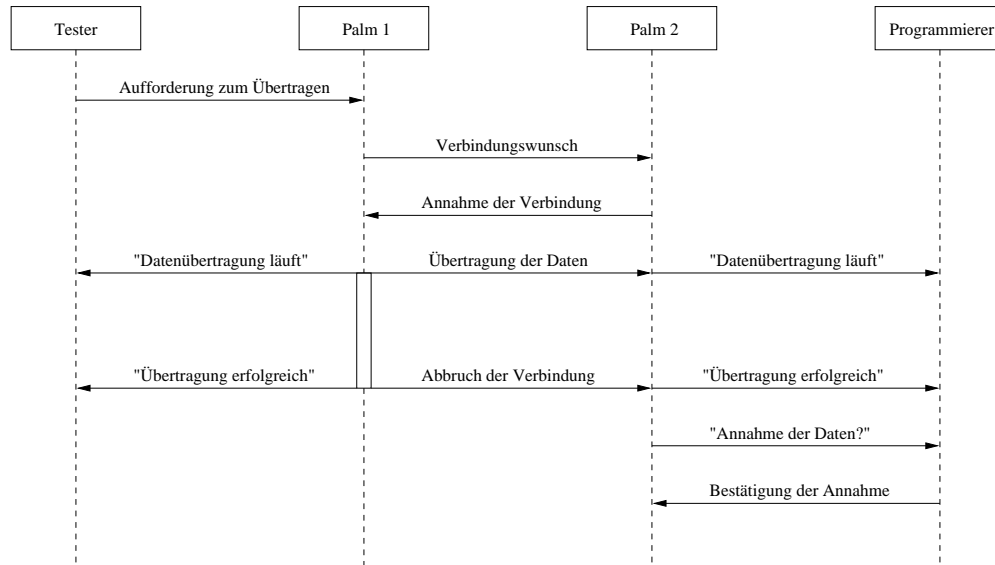


Abbildung 19: Die Kommunikation zwischen zwei Palm-Computern

hinzugefügt.

Die Direktverbindung zwischen Palm und PC funktioniert nach dem gleichen Prinzip, mit dem Unterschied, dass zusätzlich noch die Funktionsbibliothek für Verbindungen über die serielle Schnittstelle genutzt wird.

Sollen die Fehlerprotokolle per Modem als EMail an den Programmierer gesendet werden, geschieht das mittels des TCP/IP-Stacks, der seit der Version 3.1 in das PalmOS integriert ist.

Die vierte und letzte Kommunikationsvariante, bei der die Fehlerprotokolle zuerst per Hotsync-Vorgang an einen PC übertragen werden, um sie dann von dort aus weiterzuleiten, benötigt nicht die Unterstützung des Kommunikationsmanagers. Hier kommt der von Palm mitgelieferte Hotsync-Manager und eine entsprechende Desktop-Applikation zum Einsatz.

#### 4.2.5 Fazit und Ausblick

Der Palm eignet sich aufgrund seiner Fähigkeiten zur Kommunikation und seiner ständigen Verfügbarkeit und Mobilität, prinzipiell sehr gut für die Erstellung, Verteilung und Verwaltung von Fehler-Protokollen. Da allerdings größere Textmengen eingegeben werden müssen, empfiehlt sich hier die Anschaffung einer externen Tastatur.

Durch die Nutzung einer Desktop-Applikation zur Extrahierung und Umwandlung der Protokoll-Inhalte aus der Palmdatenbank auf dem PC, entspricht dieses Tool der dritten Stufe in der Einteilung der palmbasierten CASE-Tools anhand ihrer Ausstattungsmerkmale (siehe Abschnitt 3.4). Erweitert werden könnte das Programm erstens, durch die Implementation eines Editors zur Erstellung der Fehler-Protokolle auf dem PC, und deren Übertragung auf den Palm des Programmierers und zweiten, durch Hinzufügen von Formularen zur Erstellung von Mängellisten (P5).

### **4.3 Ein palmbasiertes CASE-Tool zur Unterstützung der ISO-9000 Zertifizierung (P7)**

Die ISO-9000 Norm ist ein weit verbreiteter Standard, um einen Softwareentwicklungsprozess zu bewerten (siehe Abschnitt 3.2.2). Gewöhnlicherweise beauftragen Firmen damit speziell ausgebildete Zertifizierer, oder aber es werden eigene Mitarbeiter dafür eingesetzt. Die Bewertung des Prozesses geschieht durch Beobachtung und durch Führung eines Dialoges mit allen an der Entwicklung beteiligten Personengruppen (Management, Entwicklungsabteilung, Wartungsabteilung usw.). Anhand der so gewonnenen Informationen können die 155 Fragen (eingeteilt nach einundzwanzig Kategorien) der ISO-9000 Norm jeweils positiv oder negativ beantwortet werden.

An ein palmbasiertes Tool zur Unterstützung der Zertifizierung werden folgende Anforderungen gestellt:

- Projektweise Abspeicherung aller eingegebenen Informationen,
- Auswahl der Fragen zugeordnet zu den einundzwanzig Kategorien,
- Anzeige des kompletten Wortlautes der Fragen,
- Anzeige der prozentualen Erfüllung der einzelnen Kategorien und des Gesamtergebnisses,
- Ermöglichung der Eingabe von Notizen zu den einzelnen Fragen (optional).

#### **4.3.1 Szenario Beschreibung**

Wenn der Nutzer (Zertifizierer) das Programm startet, sollte zuerst ein Bildschirm (Form) erscheinen, indem eine Liste mit den Namen aller bisher

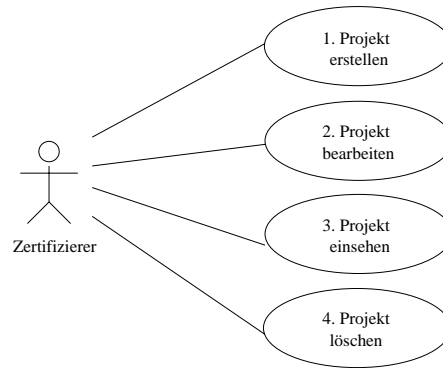


Abbildung 20: Anwendungsfälle des ISO-9000 Tools

gespeicherten Projekten angezeigt wird. Dort hat er dann die Möglichkeit zwischen den vier verschiedenen Anwendungsfällen zu wählen (siehe Abbildung 20):

- 1. Erstellung eines neuen Projektes:** Die Wahl des ersten Anwendungsfalles führt den Nutzer zu den Bildschirmen zur Erstellung eines neuen Projektes. Diese Bildschirme sollten aus Eingabemasken für die folgenden Informationen bestehen:

Nr.	Name	Typ	Beschreibung
1.	id	Integer	Eindeutige Identifizierungsnummer
2.	name	String	Name des Projektes
3.	descr	String	Kurze Beschreibung des Projektes (optional)
4.	begin	Date	Beginn des Projektes
5.	end	Date	Ende des Projektes (wenn bekannt)
6.	company	String	Name der Firma (optional)
7.	person	String	Name und Büro eines zuständigen Ansprechpartners (optional)

**(Integer:** Auf dem Palm sind Variablen vom Typ "int" zwei Byte groß.

**String:** Gemeint sind Zeichenketten vom Typ "char" mit variabler Länge.

**Date:** Das Datum wird intern durch die Anzahl von Tagen seit dem 1.1.1904 repräsentiert.)

Vom Programm können zusätzlich noch folgende Informationen hinzugefügt werden:

Nr.	Name	Typ	Beschreibung
6.	create_d	Date	Datum der Erzeugung des Projektes
7.	create_t	Time	Uhrzeit der Erzeugung des Projektes
8.	mod_d	Date	Datum der letzten Modifikation
9.	mod_t	Time	Uhrzeit der letzten Modifikation

(**Time:** Die Uhrzeit wird intern durch die Anzahl an Sekunden, die seit dem 1.1.1904 0:00 Uhr vergangen sind, repräsentiert.)

Projekt Name	
1. General	0%
2. Management responsibility	0%
3. Quality system	0%
4. Contract review	0%
5. Design control	0%
6. Document control	0%
7. Purchasing	0%
8. Purchasing-supplied products	0%
9. Product identification and traceability	0%
10. Process control	0%
11. Inspection and testing	0%

OK      Result: 0%      ▲▼

Abbildung 21: Form zur Auswahl der Kategorie

**2. Bearbeitung eines Projektes:** Ein Projekt zu bearbeiten kann erstens bedeuten bestehende Informationen zu modifizieren oder zweitens, Fragen zu beantworten. Der erste Fall führt den Zertifizierer zu den Bildschirmen aus dem Anwendungsfall "Projekt erstellen", mit dem Unterschied, dass die Eingabefelder mit den zugehörigen Informationen aus der Datenbank gefüllt sind. Der zweite Fall benötigt dagegen neue Eingabemasken. Als erstes sollte sich der Nutzer eine der einundzwanzig Kategorien aussuchen, zu der er Fragen beantworten möchte. Abbildung 21 zeigt, wie ein Bildschirm zu diesem Zweck aussehen könnte. Er besteht aus folgenden Elementen: Eine Liste mit den einzelnen Kategorien und ihrem jeweiligen Erfüllungsgrad, ein OK-Button, eine Anzeige für das Gesamtergebnat und zwei Buttons zum Hoch- und Runterscrollen in der Liste. Markiert der Zertifizierer eine Kategorie, so werden in einem weiteren Bildschirm die zugehörigen Fragen angezeigt. Hier sollte dann die Möglichkeit bestehen, z.B. mit Checkboxes, eine Antwort auszuwählen (siehe Abbildung 22).

Intern können die so gewonnenen Daten in einem 155 Elemente umfassenden Array vom Typ "Boolean" abgelegt werden.

**3. Projektdaten und -resultate einsehen:** Dieser Anwendungsfall un-

1. General: Set 4 of 7		yes	no
1. Are your employees familiar with the procedures they use?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2. Are their procedures documented?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3. Do they know where?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="button" value="OK"/> <span style="float: right;">Result: 25%</span>		<input type="button" value="Next"/>	

Abbildung 22: Form Beantwortung der Fragen

terscheidet zwischen den Möglichkeiten, der Einsichtnahme in die allgemeinen Projektinformationen und der Ermittlung und Anzeige der Resultate. Am einfachsten ist hier wohl eine Wiederverwendung der Bildschirme aus dem ersten und dem zweiten Anwendungsfall. So kann der Nutzer nach Einsichtnahme und Überprüfung der Informationen, bei eventuellen Fehlern, diese auch gleich modifizieren.

- 4. Entfernung eines Projektes:** Die Entfernung eines Projektes aus der Datenbank kann gleich im Hauptbildschirm mit der Projektliste und einem entsprechendem Button geschehen. Hat der Nutzer sich ein zu entfernendes Projekt ausgesucht und den Delete-Button gedrückt, so sollte das Programm anhand einer Sicherheitsabfrage die Richtigkeit dieser Entscheidung überprüfen, und nur bei einer positiven Bestätigung das Projekt aus der Datenbank entfernen. Danach sollte wieder der Hauptbildschirm mit aktualisierter Projektliste erscheinen.

#### 4.3.2 Die Zustände des ISO-9000 Tools

Abbildung 23 zeigt ein mögliches Zustandsdiagramm des ISO-9000 Tools. Die Zustände des Systems werden durch die verschiedenen Bildschirme (Forms) der Applikation repräsentiert und die Pfeile zeigen an, auf welchem Weg der Nutzer die Bildschirme erreichen kann. Um den Quellcode des Programmes möglichst kompakt zu halten, werden einige Forms für mehrere Anwendungsfälle genutzt. Dabei kann der Inhalt der Forms an den jeweiligen Anwendungsfall, im geringen Umfang, angepasst werden.

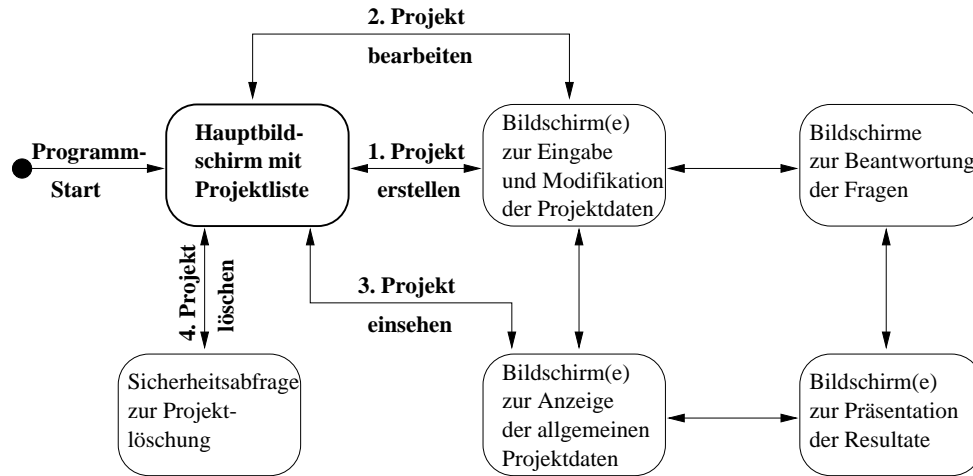


Abbildung 23: Die Zustände des ISO-9000 Tools (aus Nutzersicht)

### 4.3.3 Die Komponenten des ISO-9000 Tools

Wie schon in Abschnitt 2.5 beschrieben, sind Palm Applikationen ereignisgesteuert. Im allgemeinen besitzt dabei jede *Form* einen eigenen *EventHandler*, der die Ereignisse entgegennimmt und verarbeitet. Solche Ereignisse wären beispielsweise die Änderung von Projektdaten, die Beantwortung von Fragen oder die Anforderung von Projektergebnissen. Der *EventHandler* hat außerdem die Aufgabe alle Bildschirmelemente, wie Tabellen, Labels, Textfelder und Checkboxes zu initialisieren, indem er z.B. die Kategorien in die Tabelle schreibt oder die Fragen in den vorbereiteten Textfeldern anzeigt.

Tritt in einer *Form* ein Ereignis ein, welches Projektdaten modifiziert oder anfordert, wendet sich der *EventHandler* der entsprechenden *Form* an den *Projekthandler*, der das aktuelle Projekt verwaltet. Zu seinen Aufgaben zählt die Speicherung der Projektdaten in einer Datenstruktur, die Herausgabe und Modifizierung der Daten und die Resultatsberechnung, einmal für jede Kategorie und einmal gesamt. In den Fällen, dass geänderte oder neue Projektdaten in die Datenbank aufgenommen oder dass ein Projekt aus der Datenbank in eine Datenstruktur gelesen werden soll, nutzt der *Projekthandler* die Dienste (Funktionen) des *Datenbankmanagers*. Dieser kann Datenbanken erstellen und löschen, neue Einträge hinzufügen, die Daten von bestehenden herauslesen oder verändern und nicht mehr benötigte Projekte wieder entfernen. Die Beziehungen zwischen den einzelnen Komponenten sind in Abbildung 24 noch einmal grafisch dargestellt.

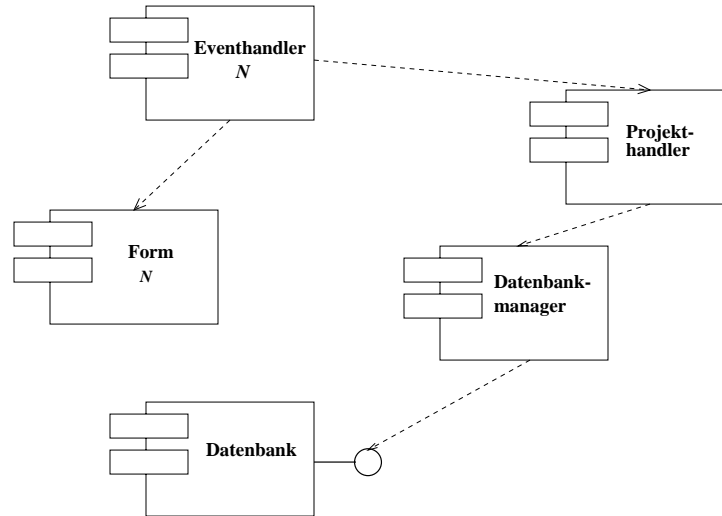


Abbildung 24: Das Komponentendiagramm des ISO-9000 Tools

#### 4.3.4 Fazit und Ausblick

Das hier entworfene Tool, zur Unterstützung der ISO-9000 Zertifizierung, entspricht der Stufe 2 in der Einteilung der palmbasierten CASE-Tools anhand ihrer Ausstattungsmerkmale (siehe Abschnitt 3.4), da alle eingegebenen Daten projektbezogen verwaltet und dauerhaft in der Palmdatenbank gespeichert werden. Eine Applikation der ersten Stufe kann in diesem Zusammenhang nicht genutzt werden, da ein Zertifizierungsprozess oft viele Wochen dauert und die Beantwortung der Fragen nur schrittweise vorstatten geht. Somit ist eine Abspeicherung der eingegebenen Daten zwingend notwendig. Weitaus denkbarer wäre hier die Nutzung der dritte Stufe, welche ein PC-Programm zum Auslesen der Datenbank und zur Unterstützung der automatischen Berichtgenerierung vorsieht. Somit müssen nicht alle Daten mühsam von Hand, beispielsweise in ein Textverarbeitungstool, übertragen werden.

### 4.4 Ein palmbasiertes CASE-Tool zur Unterstützung der CMM-Methode (P8)

Das Capability Maturity Model wurde 1991 als Referenzmodell, welches als Vergleichsnorm für Softwarelieferanten dienen sollte, veröffentlicht. Als Er-

gebnis kann mit Hilfe eines Fragebogens die Qualität eines Softwareentwicklungsprozesses in fünf Stufen eingeteilt werden (siehe Abschnitt 3.2.2). Um eine gewisse Stufe erreichen zu können, müssen dabei alle zugehörigen Fragen und alle die zu einer niedrigeren Stufe gehören, positiv beantwortet werden. Da die Methodik des Capability Maturity Models sehr ähnlich der Methodik der ISO-9000 Zertifizierung ist (Nutzung eines Fragebogens zur Bewertung eines Softwareentwicklungsprozesses), können einige Teile der Beschreibung aus Abschnitt 4.3 in den Entwurf eines palmbasierten Werkzeuges, zur Unterstützung der CMM-Methode, mit einfließen.

In folgenden Punkten differenzieren die ISO-9000 Norm und das Capability Maturity Modell laut [OG97, Seite 108]:

- *„ISO-9001 ist hauptsächlich für die Industrie gedacht, während CMM speziell für Software gilt.“*
- *„CMM ist detaillierter und spezifischer.“*
- *„ISO-9001 setzt eine annehmbare Stufe des Managements und der Prozesse des Lieferanten fest, während CMM ein Werkzeug zur Einschätzung der Leistungsfähigkeit eines Lieferanten auf einer Skala von 1 bis 5 ist.“*
- *„ISO-9001 setzt als Schwerpunkt die Beziehung zwischen Kunde und Lieferant; CMM beschäftigt sich vorrangig mit dem Prozess der Softwareentwicklung.“*

Hinzugefügt muss noch gesagt werden, dass die ISO-9000 Methode ein Assessment ist, während CMM auch Hinweise gibt, wie die jeweils nächst höhere Stufe zu erreichen ist.

Folgende Anforderungen werden an das CMM-Tool gestellt:

- Projektweise Abspeicherung aller eingegebenen Informationen,
- Zuordnung der Fragen zu den fünf Reifegradstufen,
- Anzeige des kompletten Wortlautes der Fragen,
- Anzeige der prozentualen Erfüllung der einzelnen Stufen,
- Anzeige der erreichten Stufe,
- Ermöglichung der Eingabe von Notizen zu den einzelnen Fragen (optional).

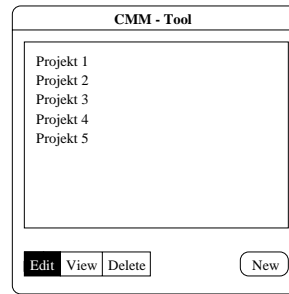


Abbildung 25: Startbildschirm der CMM-Applikation

#### 4.4.1 Szenariobeschreibung

Die Anwendungsfälle des Tools zur Unterstützung der CMM-Methode sind identisch mit denen des ISO-9000 Tools (siehe Abbildung 20). Startet der Nutzer das Programm, sollte auch hier eine Liste mit allen schon angelegten Projekten erscheinen und die Möglichkeit bestehen, mit der Markierung eines Projektes und der Nutzung eines entsprechenden Buttons, einen der Anwendungsfälle zu wählen. Abbildung 25 zeigt wie der Startbildschirm der CMM-Applikation aussehen könnte. Von dort aus hat der Nutzer die folgenden vier Möglichkeiten:

- 1. Erstellung eines neuen Projektes:** Durch die Wahl des "New"-Buttons gelangt der Nutzer wie beim ISO-9000 Tool zu den Bildschirmen zur Erstellung eines neuen Projektes, mit Eingabemasken für folgende Informationen:

Nr.	Name	Typ	Beschreibung
1.	id	Integer	Eindeutige Identifizierungsnummer
2.	name	String	Name des Projektes
3.	descr	String	Kurze Beschreibung des Projektes
4.	begin	Date	Begin des Projektes
5.	end	Date	Ende des Projektes (wenn bekannt)
6.	company	String	Name der Firma (optional)
7.	person	String	Name und Büro eines zuständigen Ansprechpartners (optional)

Dabei können vom Programm zusätzlich noch folgende Informationen hinzugefügt werden:

Projekt Name		
<input type="radio"/>	Level 1: Initial	100%
<input type="radio"/>	Level 2: Repeatable	85%
<input type="radio"/>	Level 3: Defined	0%
<input type="radio"/>	Level 4: Managed	0%
<input type="radio"/>	Level 5: Optimizing	0%
<input type="button" value="OK"/>	Level reached: 1	

Abbildung 26: Auswahlbildschirm zu den fünf Stufen (Level) der CMM-Applikation

Nr.	Name	Typ	Beschreibung
6.	create_d	Date	Datum der Erzeugung des Projektes
7.	create_t	Time	Uhrzeit der Erzeugung des Projektes
8.	mod_d	Date	Datum der letzten Modifikation
9.	mod_t	Time	Uhrzeit der letzten Modifikation

Wurden alle Informationen eingegeben, hat der Benutzer die Wahl zum Hauptbildschirm zurückzukehren oder die Fragen zu beantworten (siehe Anwendungsfall 2).

**2. Bearbeitung eines Projektes:** Der "Edit"-Button und die Wahl eines Projektes führt den Nutzer erst einmal zu den Bildschirmen mit den allgemeinen Informationen aus Anwendungsfall 1. Hier kann er die angezeigten Daten modifizieren oder zu den Bildschirmen zur Beantwortung der Fragen wechseln. Wurde die zweite Möglichkeit gewählt, erscheint eine Abfrage zur Selektierung der Reifegradstufe (siehe Abbildung 26), gefolgt von den Eingabemasken zur Beantwortung der zugehörigen Fragen (ähnlich dem ISO-9000 Tool, Abbildung 22).

**3. Projektdaten und -resultate einsehen:** Die Wahl dieses Anwendungsfalls führt zu Bildschirmen, in denen die allgemeinen Projektdaten und die erreichte Reifegradstufe angezeigt werden. Für den Fall, dass der Nutzer den Wunsch hat, die Beantwortung aller Fragen einzusehen, kann das durch eine Wiederverwendung der Bildschirme aus Anwendungsfall 2 geschehen. Zuerst erfolgt die Selektion der Reifegradstufe und danach die Anzeige der entsprechenden Fragen mit zugehöriger Antwort.

**4. Entfernung eines Projektes:** "Del" und die Wahl eines Projektes im Hauptbildschirm der Anwendung entspricht dem Anwendungsfall 4. Nach der positiven Beantwortung einer daraufhin erscheinenden Sicherheitsabfrage wird der entsprechende Datensatz aus der Datenbank gelöscht und der Projektname aus der Projektliste entfernt.

#### 4.4.2 Die Zustände und Komponenten des CMM-Tools

Aufgrund der Ähnlichkeit zwischen dem ISO-9000 Tool und dem CMM-Tool ist das Zustandsdiagramm aus Abbildung 23 auch hier uneingeschränkt gültig. Das Diagramm beschreibt die Zustände des Programms anhand der verschiedenen Bildschirme und der Wege, auf denen diese zu erreichen sind. Die Benennung und Beschreibung der Komponenten des ISO-9000 Tools und ihre Beziehungen untereinander (siehe Abschnitt 4.3.3), kann aus den selben Gründen gleichfalls übernommen werden.

#### 4.4.3 Fazit und Ausblick

Die Unterschiede zwischen den Programmen zur Bewertung eines Softwareprozesses nach der ISO-9000 Norm und nach dem Capability Maturity Model, beschränken sich größtenteils auf die Gestaltung einzelner Bildschirme (z.B. bei der Wahl der Kategorien oder Stufen und bei der Auswertung) und auf die Speicherung und Auswertung der eingegebenen Daten. Alle anderen Programmkomponenten können fast vollständig übernommen werden.

Mit den in diesem Entwurf beschriebenen Features entspricht das CMM-Tool der zweiten Stufe bei der Einteilung nach Ausstattungsmerkmalen (siehe Abschnitt 3.4). Aufgewertet werden könnte das Programm durch eine PC-Applikation, welche die Datenbank analysiert und aus den gespeicherten Daten automatisch ein Bericht generiert, in dem beispielsweise, neben den allgemeinen Projektinformationen, alle gestellten Fragen, mit den gegebenen Antworten, tabellarisch aufgeführt werden. Darüber hinaus kann hierbei ein CMM-Bewertungsvergleich mehrerer Projekte vorgenommen werden.

## 5 Spezifikation, Entwurf und Implementation eines palmbasierten Function–Point Zähltools

Der Grundgedanke der Function–Point–Analyse (FPA) liegt in der Untersuchung einer Produktbeschreibung (z.B.: Spezifikation) auf ihren "funktionalen Gehalt" und damit verbunden, in ihre Einteilung in folgende diskrete Komponenten (Funktionstypen):

- Eingaben (EI),
- Ausgaben (EO),
- Abfragen (EQ),
- Interne Datenbestände (ILF) und
- Referenzen auf externe Datenbestände (EIF).

Nach der Analyse der Produktbeschreibung erfolgt eine Einteilung der Komponenten in "*einfach*", "*mittel*" und "*komplex*", anhand der Anzahl von Daten- und Kontrollelementen (Data Element Types (DET) und Record Element Types (RET)) und eine Wichtung mit folgenden Faktoren:

Komponente	einfach	mittel	komplex
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Durch die Summierung der Ergebnisse erhält man die *Unadjusted Function–Points* (UFP).

Anhand von Erfahrungen und Resultaten von ähnlichen, schon abgeschlossenen Projekten, kann mit dieser Zahl eine Schätzung des Aufwandes vorgenommen werden.

Eine Erweiterung der Function–Point Analyse ist die Full–Function–Point

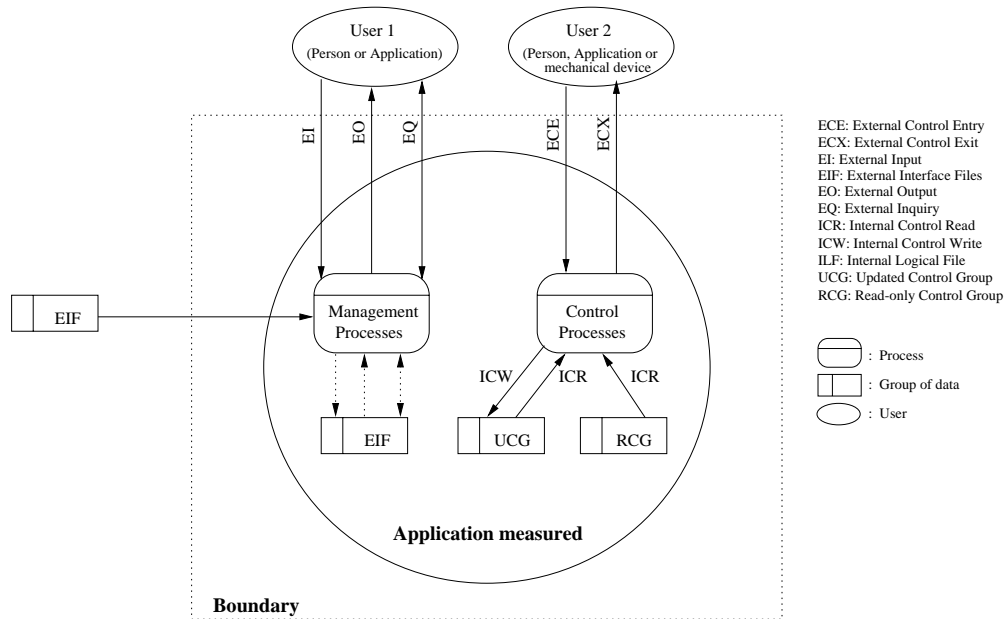


Abbildung 27: Die Funktionstypen der FFP-Methode nach IFPUG

Methode ([SPMA<sup>+</sup>97]). Sie dient der Aufwandsschätzung von Softwareprojekten für Embedded-Systeme und enthält neben den oben genannten Komponenten (Management-Funktionstypen) noch weitere Kontroll-Funktionstypen (siehe Abbildung 27), mit eigenen Berechnungsvorschriften. An einer endgültigen Version (Version 2.0) der FFP-Methode wird z.Z. noch durch das “*CO*mmun Software Measurment International Consortium (*COSMIC*<sup>30</sup>)” gearbeitet (siehe [Abr99] und [ADS<sup>+</sup>99]). Aus diesem Grund beziehe ich mich bei der Umsetzung des Function-Point CASE-Tools auf die Version 1.0 der FFP-Methode.

Eine komplette Beschreibung der Function-Point Analyse und der Full-Function-Point Methode befindet sich in [BF00] und [DA99, ab Seite 247] vergleicht die beiden Methoden anhand eines Experimentes.

## 5.1 Computergestützte Eingabe der Daten

Es sind zwei Möglichkeiten vorstellbar, wie ein Function-Point Zähler die ermittelten Daten in eine entsprechende Applikation eingeben könnte. Die

<sup>30</sup>siehe [Sym99]

		# of Data Element Types		
		1-19	20-50	>50
# of Record Element Types	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	2-5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	>6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Abbildung 28: Die Komplexitätsmatrix einer Funktion vom Typ "EIF"

erste Möglichkeit ("detailed") geht davon aus, dass bei jedem Auffinden einer Komponente im zu analysierendem Dokument ein Eintrag in das Programm vorgenommen werden soll, wobei die Einteilung in "einfach", "mittel" und "komplex" anhand der Anzahl der DETs und RETs, Aufgabe der Applikation ist. Zu diesem Zweck bietet sich die sogenannte Komplexitätsmatrix an (siehe Abbildung 28). Der Nutzer wählt zuerst den Typ der Komponente aus und entscheidet sich dann, anhand der Anzahl der Datenelemente, für eine der Checkboxen. Der Computer kann jetzt die Komplexitätsgrad ermitteln und den Wert abspeichern. Danach fährt der Nutzer mit der nächsten Komponente fort oder er beendet die Eingabe.

Bei der zweiten Möglichkeit ("summary") wird die Anzahl der einzelnen Funktionstypen, zugeordnet zu den Komplexitätsgraden, direkt in eine Tabelle übertragen (siehe Abbildung 29). Das setzt aber voraus, dass die Werte schon irgendwo zusammengefasst gesammelt wurden. Diese Methode erfordert einen größeren manuellen Aufwand und eine größere Sorgfalt vom Nutzer, dafür steigt aber die Zählgeschwindigkeit.

Welche der beiden Methoden genutzt wird, ist letztendlich von den Vorlieben des Nutzers und von seinem Erfahrungsstand abhängig.

## 5.2 Die Anforderungen an das Function-Point Zähltool

An das Function-Point Zähltool wurden folgende Anforderungen gestellt:

1. Zu unterstützen ist die Eingabe der Management-Funktionstypen der FPA-Methode und der zusätzlichen Kontroll-Funktionstypen der FFP-Methode, als Grundlage zur Berechnung der Function-Points.

	<b>low</b>	<b>average</b>	<b>high</b>	<b>#</b>
<b>ILFs</b>	42	14	4	60
<b>EIFs</b>	23	18	3	44
<b>EIs</b>	85	24	32	141
<b>EOs</b>	65	78	9	152
<b>EQs</b>	22	13	10	45
<b>#</b>	237	147	58	442

Abbildung 29: Eine Wertetabelle für die FPA Methode

2. Die Eingabe ist in Form von Komplexitätsmatrizen und Wertetabellen zu ermöglichen.
3. Alle eingegebenen Informationen sind in einer Datenbank zu sichern.
4. Eine spätere Erweiterung, Modifikation und Anzeige aller abgespeicherten Informationen ist zu ermöglichen.
5. Alle Informationen sind projektweise abzuspeichern.
6. Zur besseren Übersicht und zum späteren Vergleich, sind Projekte in Phasen (oder Versionen, je nach Anwendungsfall), diese in Applikationen und diese in Prozesse zu unterteilen (siehe Abschnitt 5.3).
7. Nur zu den einzelnen Prozessen werden die zur Function-Point Berechnung relevanten Daten eingegeben und abgespeichert.
8. Folgende allgemeine Informationen sind für jedes Projekt einzugeben und abzuspeichern:
  - Name des Projektes,
  - Kurze Beschreibung,
  - Beginn und Ende des Projektes (Ende nur wenn bekannt),
  - Kosten des Projektes.
9. Folgende allgemeine Informationen sind für jede Phase (Version) einzugeben und abzuspeichern:
  - Namen,

- Kurze Beschreibung,
  - Beginn und Ende (Ende nur wenn bekannt),
  - Kosten.
10. Folgende allgemeine Informationen sind für jede Applikation einzugeben und abzuspeichern:
- Name der Applikation,
  - Kurze Beschreibung,
  - Beginn und Ende der Applikationsentwicklung (Ende nur wenn bekannt),
  - Kosten der Applikation,
  - Name des zu analysierenden Dokumentes,
  - Name des Zählers.
11. Folgende allgemeine Informationen sind für jeden Prozess einzugeben und abzuspeichern:
- Name des Prozesses,
  - Abschnittsnummer im zu analysierenden Dokument,
  - Anzahl der Seiten der Prozessbeschreibung.
12. Die Auswertung eines Zählprojektes hat aus folgenden Angaben zu bestehen:
- Anzahl der Function-Points der Management-Funktionstypen (FPA), der Kontroll-Funktionstypen (FFP) und das Gesamtergebnis (UFP),
  - Anzahl der Function-Points, zugeordnet zu den einzelnen Funktionstypen,
  - Summe der analysierten Seiten,
  - Anzahl der Prozesse,
  - Durchschnittliche Anzahl der analysierten Seiten pro Prozess,
  - Durchschnittliche Anzahl von Function-Points pro Seite,
  - Durchschnittliche Kosten pro Function-Point und

- Dauer des Projektes.
13. Die oben genannten Resultate der Auswertung sind soweit es möglich ist, auch für jede Phase (Version), für jede Applikation und für jeden Prozess bereitzustellen.
  14. Die Datenbank ist bei Hotsync-Vorgängen auf dem Desktop zu sichern.
  15. Es ist eine Desktop-Applikation zu entwickeln, welche die Daten aus gesicherten Datenbanken extrahiert und daraus automatisch Berichte und Statistiken generiert, welche von gängigen Textverarbeitungsprogrammen und Tabellenkalkulationen verarbeitet werden können (z.B.: Dokumente und Tabellen in Form von ASCII-Texten).

### 5.3 Die Datenbank des Function-Point Zähltools

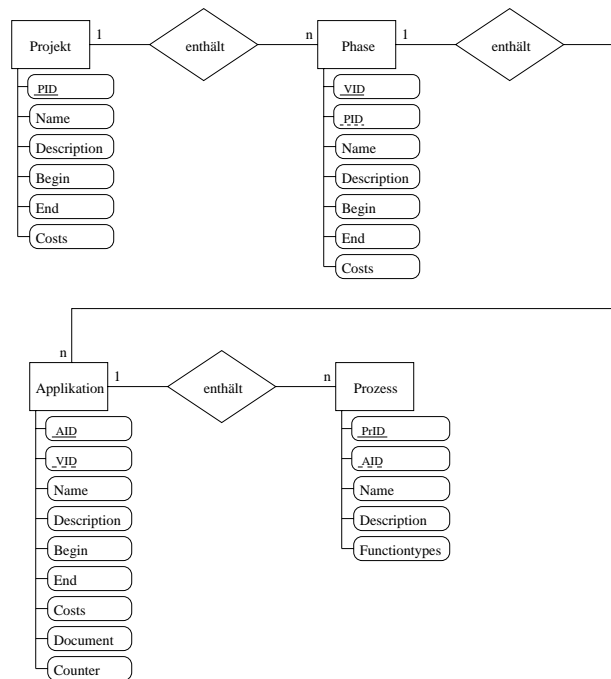


Abbildung 30: Das ER-Modell der FPC-Datenbank

Wie in den Anforderungen beschrieben, wird in "FPC" ein Projekt hierarchisch aus Phasen (oder Versionen), Applikationen und Prozessen aufgebaut.

<b>Konfigurations - Record</b> ( <i>struct ConfRec</i> )
<b>Projekt - Record 1</b> ( <i>struct ProjRec</i> )
<b>Projekt - Record 2</b>
⋮
<b>Projekt - Record n</b>
<b>Phasen - Record 1</b> ( <i>struct VerRec</i> )
<b>Phasen - Record 2</b>
⋮
<b>Phasen - Record n</b>
<b>Applikations - Record 1</b> ( <i>struct AppRec</i> )
<b>Applikations - Record 2</b>
⋮
<b>Applikations - Record n</b>
<b>Prozess - Record 1</b> ( <i>struct ProcRec</i> )
<b>Prozess - Record 2</b>
⋮
<b>Prozess - Record n</b>

Abbildung 31: Die Reihenfolge der Records in der FPC-Datenbank

Ein *Prozess* ist ein funktioneller Teil einer komplexen *Applikation* und entspricht der "counting boundary" der Function-Point Zählmethode. Eine Applikation besteht aus einer bestimmten Anzahl von Prozessen und gehört in FPC immer zu einem *Softwareprojekt*. Dadurch wird es ermöglicht, auf einem Palm mehrere Softwareprojekte gleichzeitig zu verwalten und zu vergleichen. In seiner Entstehung durchläuft ein Softwareprodukt mehrere *Phasen*, wobei verschiedene Dokumente den jeweils aktuellen Entwicklungsstand präsentieren (z.B.: Entwurfsdokumente, Spezifikationen, Beschreibungen des fertigen Produktes oder von verschiedenen Produktversionen). Diese Dokumente werden häufig einzeln analysiert, um den Entstehungsprozess besser verfolgen zu können. In FPC entspricht die Version den verschiedenen Phasen und Produktversionen des Softwareprojektes.

Die Vorteile dieser Hierarchie liegen in der besseren Übersichtlichkeit bei Projekten mit einem großen Umfang und hoher Komplexität und in der Möglichkeit, die Anzahl der Function-Points und die ermittelten statistischen Werte der verschiedenen Phasen und Applikationen, untereinander vergleichen zu können.

Abbildung 30 zeigt das Entity-Relationship-Modell der FPC-Datenbank. Allerdings sind Palm-Datenbanken nicht relational sondern, wie in Abschnitt 2.5 beschrieben, recordbasiert aufgebaut.

Die Reihenfolge der Records in der FPC-Datenbank ist in Abbildung 31 dargestellt. Sie enthält nacheinander den Konfigurationsrecord mit allgemeinen Informationen zur Datenbank, alle Projekte, alle Phasen, alle Applikationen und alle Prozesse.

Die einzelnen Recordtypen haben folgende Deklaration:

**Konfiguration:** struct ConfRec {UInt FirstProj, NumProj,  
FirstStat, NumStat, FirstApp, NumApp,  
FirstProc, NumProc, LastID;};

**Project:** struct ProjRec {UInt id; Word type, str1len, str2len;  
ULong Begin, End; Long costs;  
UInt Nr, Fst, Lst, Nxt;}

**Phase:** struct VerRec {UInt id; Word type, str1len, str2len;  
ULong Begin, End; Long costs;  
UInt Nr, Fst, Lst, Nxt, Up;}

**Applikation:** struct AppRec {UInt id; Word type, str1len, str2len;  
ULong Begin, End; Long costs;  
UInt Nr, Fst, Lst, Nxt, Up;  
UInt counternamelen, docnamelen;}

**Prozess:** struct ProcRec {UInt id; Word type; UInt Nxt, Up;  
Word str1len, str2len; UInt NrPages;  
struct FPADData FPA; struct FFPData FFP;}

<b>Element</b>	<b>Beschreibung</b>
FirstProj	Index des ersten Projektes in der Datenbank
NumProj	Anzahl an Projekten in der Datenbank
FirstStat	Index der ersten Phase in der Datenbank
NumStat	Anzahl an Phasen in der Datenbank
FirstApp	Index der ersten Applikation in der Datenbank
NumApp	Anzahl an Applikation in der Datenbank
FirstProc	Index des ersten Prozesses in der Datenbank
NumProc	Anzahl an Prozessen in der Datenbank
LastID	Die höchste in der Datenbank verwendete ID
ID	Eindeutige ID des Records
type	Typ des Records (Projekt, Phase, Applikation, Prozess)
str1len	Länge des Recordnamens in Byte
str2len	Länge der Kurzbeschreibung in Byte
Begin	Start des Projektes, der Phase, der Applikation
End	Ende des Projektes, der Phase, der Applikations
costs	Kosten des Projektes, der Phase, der Applikation
NrPages	Anzahl der Seiten der Prozessbeschreibung
counternamelen	Länge des Namens des Counters in Byte
docnamelen	Länge des Namens des Dokumentes in Byte
FPADData	Anzahl der einzelnen Management-Funktionstypen
FFPData	Anzahl der einzelnen Kontroll-Funktionstypen
Nr	Projekt: Anzahl an zugehörigen Phasen Phase: Anzahl an zugehörigen Applikationen Applikation: Anzahl an zugehörigen Prozessen
Fst	Projekt: ID der ersten zugehörigen Phase Phase: ID der ersten zugehörigen Applikation Applikation: ID des ersten zugehörigen Prozesses
Lst	Projekt: ID der letzten zugehörigen Phase Phase: ID der letzten zugehörigen Applikation Applikation: ID des letzten zugehörigen Prozesses
Nxt	ID des nächsten Records vom gleichen Typ
Up	Phase: ID der zugehörigen Phase Applikation: ID des letzten zugehörigen Prozesses Prozess: ID der zugehörigen Applikation

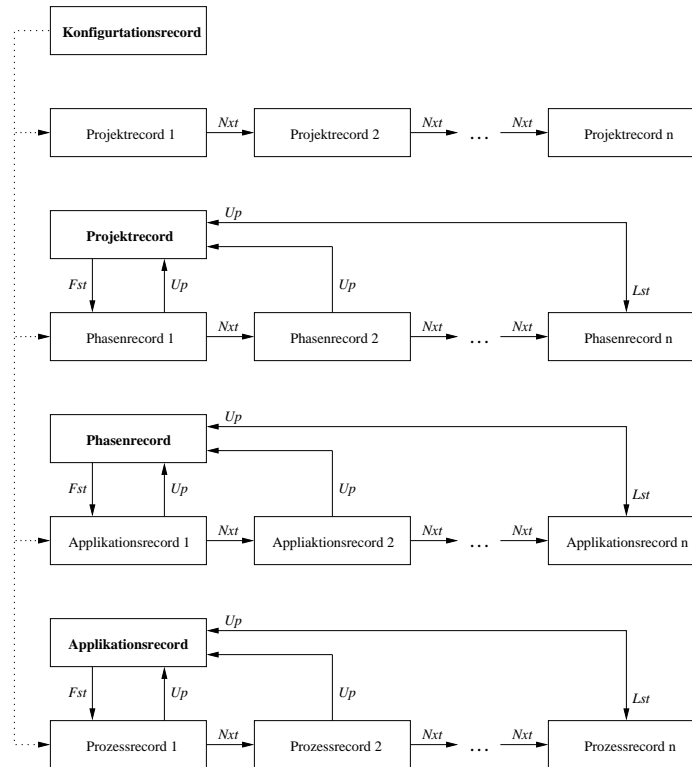


Abbildung 32: Die Verknüpfungen der Records der FPC-Datenbank

Die Abbildung 32 stellt dar, wie die einzelnen Einträge in der FPC-Datenbank miteinander verknüpft sind. Elemente des Konfigurationsrecord zeigen auf das erste Projekt, auf die erste Phase, auf die erste Applikation und auf den ersten Prozess in der FPC-Datenbank. Jedes Projekt zeigt auf die erste und letzte zugehörige Phase und auf das nächste Projekt. Jede Phase zeigt auf die erste und letzte zugehörige Applikation, auf das zugehörige Projekt und auf die nächste Phase des gleichen Projektes. Jede Applikation zeigt auf den ersten und letzten zugehörigen Prozess, auf die zugehörige Phase und auf die nächste Applikation der gleichen Phase. Jeder Prozess zeigt auf die zugehörige Applikation und auf den nächsten Prozess der gleichen Applikation.

Mit diesen Informationen wird es den Funktionen des Datenbankmanagers ermöglicht, durch die FPC-Datenbank zu navigieren. Die Links der Datenbankeinträge enthalten dabei immer die ID und nicht den Index des Zielrecords, da sonst im schlimmsten Fall bei der Entfernung oder Erzeugung

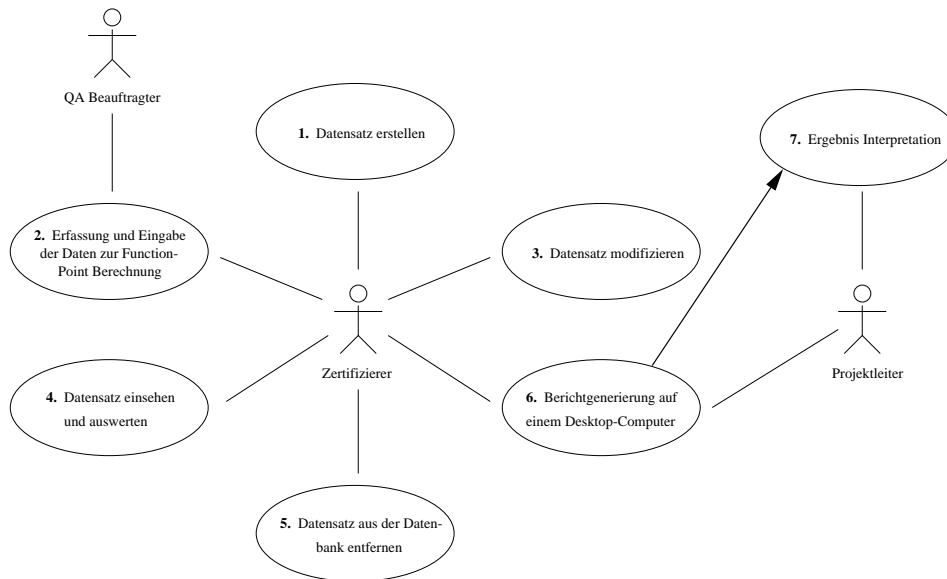


Abbildung 33: Das Anwendungsfalldiagramm von FPC

eines Eintrages, alle Verweise in allen Records angepasst werden müssten. Der Nachteil ist allerdings, dass immer eine Funktion zum Umsetzen der IDs in Indizes ausgeführt werden muss, wenn der Datenbankmanager auf ein Record in der Datenbank zugreifen will. Damit dies so effizient wie möglich geschehen kann, sind alle Einträge mit steigender ID-Nummer in der Datenbank angeordnet. Somit kann der Algorithmus zur Binären Suche (siehe [Sed94]) genutzt werden.

## 5.4 Szenariobeschreibung

Nachdem die Vorgaben aufgeschrieben und der u.a. daraus resultierende allgemeine Aufbau der Datenbank geklärt ist, folgt als nächste Aufgabe, der Entwurf der grundsätzlichen Bedienstruktur des Function-Point Zählprogramms. Dazu muss erst einmal analysiert werden, welche Personen (Akteure) an dem Zertifizierungsprozess beteiligt sind und welche Anwendungsfälle sich daraus ergeben. Das Resultat wird Abbildung 33 aufgezeigt.

Der **Zertifizierer** untersucht Projektdokumente auf ihren "funktionellen Gehalt" und zählt die Anzahl an Funktions- und Datentypen. Mit dem Function-Point Zähltool kann er Datenbankeinträge anlegen, modifizieren und löschen, er kann die ermittelten Daten eingeben und sich die Ergebnis-

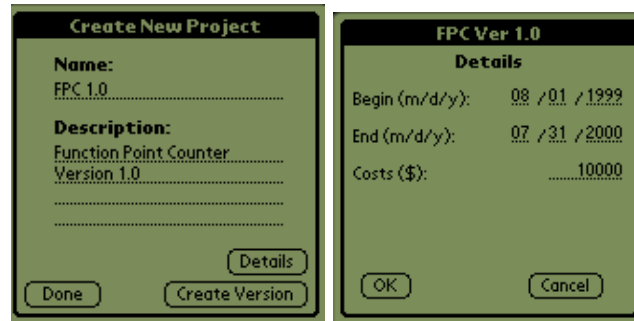


Abbildung 34: Die Bildschirme zur Erstellung eines neuen Projektes

se anzeigen lassen und überprüfen. Ist die Untersuchung beendet, werden die Daten an einen Desktop-Computer gesendet, um dort mit den Zählergebnissen ein Bericht zu erstellen, welcher dem zuständigen **Projektleiter** übergeben wird. Dieser kann mit den dort gemachten Aussagen Abschätzungen über den Projektverlauf und die benötigten Ressourcen tätigen. Sollte der Zertifizierer Fragen bzgl. des Projektes haben, deren Klärung notwendig für den Zertifizierungs-Prozess ist, kann er diese dem **QA-Beauftragten** des Projektes stellen.

Der **erste Anwendungsfall** beschäftigt sich mit der Erstellung eines neuen Datensatzes. Zu den entsprechenden Bildschirmen gelangt der Nutzer (Zertifizierer) nach dem Start der Palmapplikation, durch den Button *“Create New Projekt”*. Hier können der Name und die Kurzbeschreibung des Softwareprojektes sowie unter *“Details”*, die Informationen über den zugehörigen Zeitraum und den Kosten eingegeben werden (siehe Abbildung 34). Menüpunkte bieten an, den Datensatz zu speichern oder zum Startbildschirm zurückzukehren. Wurden alle gewünschten Daten eingegeben, so gelangt man durch den *“OK”*-Button wieder zum Startbildschirm oder durch den *“Create Version”*-Button, zu den Bildschirmen zur Erstellung eines neuen Phasendatensatzes. In beiden Fällen werden die eingegebenen Informationen der FPC-Datenbank hinzugefügt.

Die Bildschirme zur Erstellung eines neuen Phasendatensatzes entsprechen weitestgehend den Bildschirmen zur Erstellung eines neuen Zählprojektes. Wurden auch hier alle gewünschten Daten eingegeben, so gelangt man durch den *“OK”*-Button zurück zum aktuellen Projektdatensatz und kann eine weitere Phase erstellen. Mit *“Create Application”* gelangt man zu den Bildschirmen zur Erstellung eines neuen Applikationsdatensatzes. Auch in diesen Bildschirm kann der Name, eine Beschreibung, der Beginn und das Ende

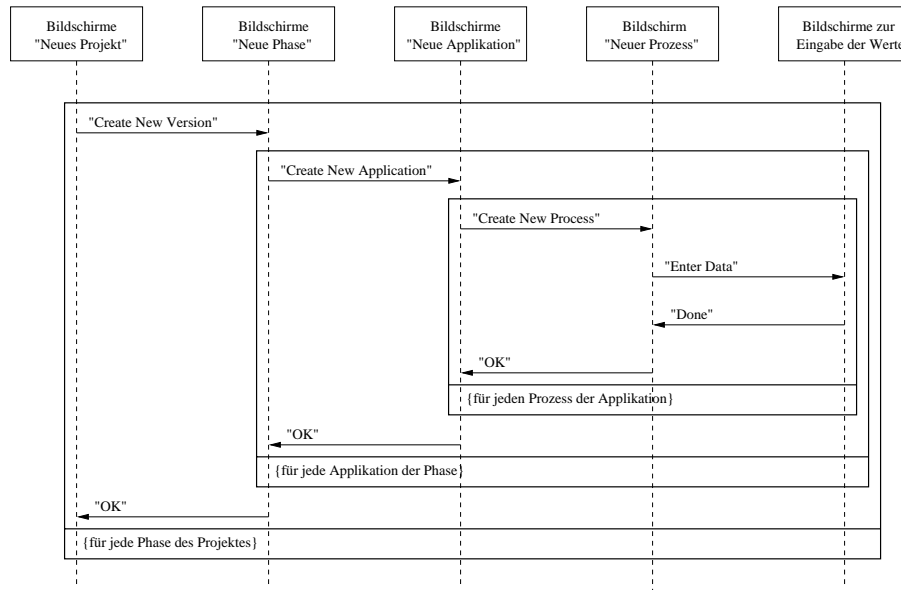


Abbildung 35: Das Sequenzdiagramm zur Erstellung neuer Datensätze

sowie die Kosten eingegeben werden. Zusätzlich dazu existieren unter “*Details*” Eingabemöglichkeiten für den Name des Counters und den Namen des zu analysierenden Dokumentes. “*OK*” und “*Create*” *Process* schließen, nach Speicherung der neuen Informationen, die Bildschirme zur Erstellung eines neuen Applikationsdatensatzes und man gelangt entweder zurück zum aktuellen Phasendatensatz oder zum Bildschirm zur Erstellung eines neuen Prozessdatensatzes.

Der Bildschirm zur Erstellung eines Prozessdatensatzes unterscheidet sich größtenteils von den vorhergegangenen. Er enthält ein Eingabefeld für den Namen des Prozesses, eins für die Abschnittsnummer im analysierten Dokument und eins für die Anzahl der Seiten, die den Prozess beschreiben (für statistische Berechnungen). Mit den Auswahlfeldern “*Counting Method*” und “*Inputtype*” können für die folgende Eingabe der Daten, zu den Funktionstypen, noch Einstellungen getroffen werden. Die erste Wahlmöglichkeit entscheidet über die Art der Funktionstypen, zu denen Daten eingegeben werden sollen (Management-Funktionstypen oder Kontroll-Funktionstypen) und die zweite Wahlmöglichkeit entscheidet über die Art der Eingabe der Daten (“*detailed*” oder “*summary*”). Mit dem Button “*Enter Data*” gelangt man zur Eingabe der Anzahl der verschiedenen Funktionstypen (siehe Anwendungsfall 2) und “*Done*” schließt den Bildschirm, speichert den Prozess

in der Datenbank und führt zum aktuellen Applikationsdatensatz zurück. Zum besseren Verständnis wurde die Bedienungsstrategie zur Erstellung von neuen Datensätzen noch einmal in einem Sequenzdiagramm grafisch dargestellt (siehe Abbildung 35). Die Objekte entsprechen den Bildschirmen bzw. den dahinterliegenden Funktionen und Datenstrukturen und die Nachrichten entsprechen den Ereignissen, die zum Aufruf des nächsten oder vorherigen Bildschirmes führen (Drücken des entsprechenden aufgeführten Buttons). Beim **zweiten Anwendungsfall** geht es um die Eingabe der Daten zur Berechnung der Function-Points. Wie schon in Abschnitt 5.1 beschrieben, existieren dabei zwei Möglichkeiten (*“detailed”* oder *“summary”*), welche sich wiederum in den Eingabemasken für die Management-Funktionstypen und für die Kontroll-Funktionstypen unterscheiden. Abbildung 36 zeigt Beispiele

	1-4	5-15	>15
0-1:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
>2:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	l	a	h	#
ILFs:	2	0	1	3
EIfs:	3	1	0	4
Els:	1	0	3	4
EO:s:	4	2	2	8
EQ:s:	6	3	0	9

Abbildung 36: Beispiel für eine Komplexitätsmatrix und eine Wertetabelle in FPC

für eine Komplexitätsmatrix und eine Wertetabelle zur Eingabe der Anzahl von Management-Funktionstypen. Im linken Bildschirm kann sich der Nutzer eine passende Checkbox aussuchen und mit *“OK”* bestätigen oder mit *“Cancel”* die Eingabe abbrechen. Im rechten Bildschirm wählt der Nutzer sich ein Eintrag in der Tabelle aus den er ändern möchte, woraufhin ein Bildschirm zur Eingabe des neuen Wertes erscheint. Nach dessen Bestätigung wird die Tabelle angepasst und der Nutzer kann mit einem anderen Eintrag fortfahren oder zum vorherigen Bildschirm zurückkehren.

Die Eingabemasken für die Kontroll-Funktionstypen differenzieren zwar im Aufbau und Aussehen, die Bedienstrategie bleibt aber die selbe.

Den **dritten** und den **fünften Anwendungsfall** erreicht der Nutzer durch den *“Modify Project”*-Button im Startbildschirm. Um einzelne Datensätze gezielt modifizieren oder löschen zu können, muss eine Möglichkeiten existieren, durch die FPC-Datenbank zu *“browsen”*. Diese bieten die in Ab-



Abbildung 37: Beispielbildschirme zum "Browsen" durch die FPC-Datenbank

bildung 37 dargestellten Bildschirme. Der Bildschirm ganz links besteht aus einer Liste mit allen vorhandenen Projekten, einer Buttonreihe und einem "OK"-Button. Hier hat der Nutzer nun mehrere Möglichkeiten:

1. "Del" und die Wahl eines Eintrages in der Liste, löscht nach einer Sicherheitsabfrage das entsprechende Projekt, mit all seinen Phasen-, Applikations- und Prozessdatensätzen in der FPC-Datenbank (Anwendungsfall 5).
2. "Info" und die Wahl eines Eintrages in der Liste, ermöglicht die Modifikation des Projektdatensatzes und damit auch die Erzeugung weiterer Phasen-, Applikations- und Prozessdatensätze (Anwendungsfall 3), durch die Anzeige der aktuellen Projektdaten in den schon bekannten Bildschirmen aus dem ersten Anwendungsfall. Mit "OK" gelangt man von dort zurück zur Übersicht.
3. Durch "Ver's" und die Wahl eines Listeneintrages, werden in der Liste alle dem Projekt zugehörigen Phasen angezeigt (siehe rechter Bildschirm in Abbildung 37).
4. Mit dem "OK"-Button gelangt man zum Startbildschirm zurück.

Die dritte Möglichkeit dient zur Navigation durch die Struktur der FPC-Datenbank. Von den Projektdatensätzen gelangt man so zu den Phasendatensätzen des ausgewählten Projektes. Jetzt hat man die Möglichkeit einen Phasendatensatz inklusive aller zugehörigen Applikations- und Prozessdatensätze zu löschen oder man modifiziert die Daten einer bestimmten Phase und kann dann auch weitere Applikationen und Prozesse erstellen. Oder

man kann sich mit dem Button “*App’s*”, alle zu einer Phase gehörenden Applikationen anzeigen lassen. Mit dem “*OK*”-Button gelangt man wieder zur Projektübersicht. So kann man mit dem Navigationsbildschirm auf alle Datensätze gezielt zugreifen. Mit den Buttons *Ver’s*, “*App’s*” und “*Proc’s*” gelangt man jeweils zum nächsten Datensatztyp und mit *OK* wieder zum vorherigen Bildschirm zurück.

Zur Besichtigung und Auswertung der Datensatzinformationen (**Anwendungsfall 4**) gelangt man durch den “*Analyze Project*”-Button. Hier erscheint der schon aus dem dritten und fünften Anwendungsfall bekannte Navigationsbildschirm, mit dem einen Unterschied, dass der “*FPS*”-Button den “*Del*”-Button ersetzt hat (siehe Abbildung 38). Wurde der “*FPS*”-Button



Abbildung 38: Bildschirm zur Auswahl des zu analysierenden Datensatzes

ausgewählt und eins der Datensätze aus der Liste, dann werden die Zahlen der verschiedenen Funktionstypen zusammengezählt, die Function-Points berechnet und es erscheint der in Abbildung 39 (links) dargestellte Bildschirm. In der ersten Zeile stehen die Function-Points der Management-

FPC Ver 1.0	
Project	
	FP
FPA:	219
FFP:	88
UAF:	307

Analyze OK

FPC Ver 1.0	
Project	
Pages:	12
Processes:	6
Pages/Proc:	2
FPS/Page:	57
Costs/FP:	14
Duration (days)	365

OK

Abbildung 39: Bildschirme zur Auswertung der Datensätze

Funktionstypen, in der zweiten Zeile die Function-Points der Kontroll-Funktionstypen und in der letzten die Anzahl der Unadjusted Function-Points. Wird einer der beiden oberen Werte angeklickt, so werden die Function-Points aufgeschlüsselt zu den zugehörigen Funktionstypen angezeigt.

Durch die Wahl des “*Analyze*”-Buttons kann man sich noch weitere statistische Werte berechnen lassen (siehe rechter Bildschirm in Abbildung 39). Das sind im einzelnen:

- Die Anzahl der gezählten Seiten (Pages).
- Die Anzahl der zugehörigen Prozesse (Processes).
- Die durchschnittliche Anzahl von Seiten pro Prozess (Pages/Proc).
- Die durchschnittliche Anzahl von Function-Points pro Seite (FPs/Page).
- Die durchschnittlichen Kosten pro Function-Point (Costs/FP). Dieser Wert ist wichtig um die Kosten für nachfolgende Projekte abschätzen zu können.
- Die Dauer des Projektes, der Version oder der Applikation in Tagen.

Zur Ansicht der gespeicherten Informationen der Datensätze der FPC-Datenbank gelangt man durch den “*Info*”-Button und die Wahl eines Eintrages in der entsprechenden Liste im Navigationsbildschirm.

Der **sechste Anwendungsfall** wird durch eine eigenständige Applikation für Desktop-PCs, mit dem MS Windows 9x- oder MS Windows NT-Betriebssystem, bedient (“*FPCAnalyze.exe*”). Sie hat die Aufgabe Zählprojekte auszuwerten und daraus ein Bericht zu generieren. Dabei wird eine FPC-Datenbank, welche durch den Hotsync-Vorgang automatisch auf dem PC gesichert wird, eingelesen und nach vorhandenen Zählprojekten durchsucht. Nach Auswahl eines Projektes können die folgenden drei Dateitypen aus den Daten des Projektes erzeugt werden:

1. Die erste Datei (“*Projektname.dat*”) enthält die eingegebenen Informationen aller Datensätze des Zählprojektes. Das schließt die Zahlen der Funktionstypen bei den Prozessen mit ein.

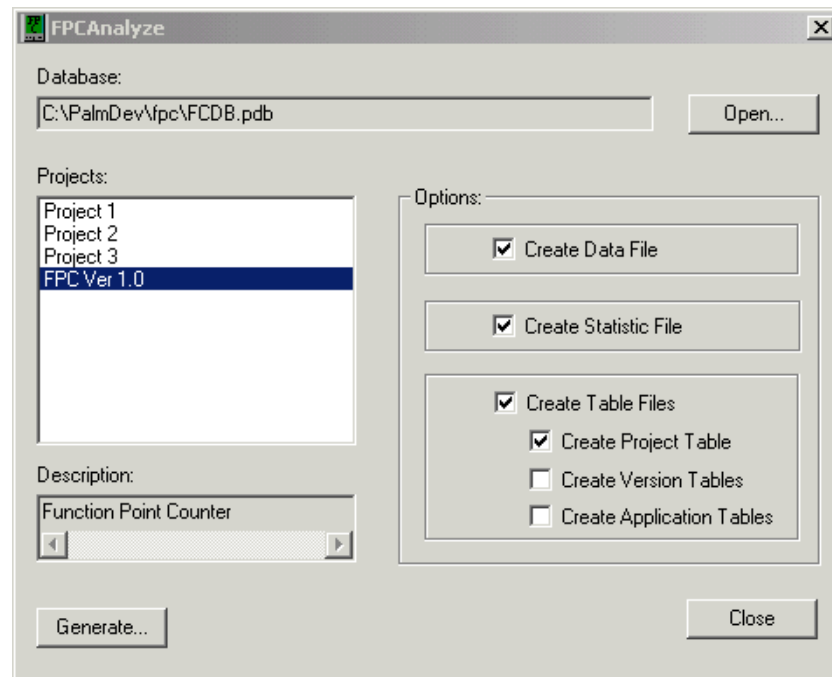


Abbildung 40: Windowsapplikation zur Auswertung und Berichtgenerierung

2. Die zweite Datei ("*Projektname.sta*") enthält die Function-Points der Funktionstypen und statistische Berechnungen zu den einzelnen Datensätzen.
3. "*Tables*" ist ein Verzeichnis, welches Tabellen mit den Zahlen der verschiedenen Funktionstypen und den ermittelten Function-Points zu den Datensätzen Projekt, Version, und Applikation enthält ("*Projektname.tab*", "*Phasenname.tab*", "*Applikationsname.tab*"). Jede Datei enthält eine Tabelle und trägt den Namen des Projektes, der Phase oder der Applikation. Diese Dateien können in alle gängigen Tabellenkalkulationsprogramme importiert werden.

Nach dem Start der Applikation erscheint das in Abbildung 40 dargestellte Fenster, welches folgendermaßen aufgeteilt ist: Im oberen Textfeld steht die Position der FPC-Datenbank, welche mit dem "Open"-Button und dem daraufhin erscheinenden Dateidialogfeld ausgewählt wird. Den linken mittleren Teil nimmt eine Liste zur Anzeige der in der Datenbank abgespeicherten Zählprojekte sowie ein Feld mit der Beschreibung des ausgewählten Projektes ein. Auf der rechten Seite des Applikationsfensters kann mit den Checkboxes

entschieden werden, welche Berichtdateien erstellt werden sollen und ganz unten befinden sich die Buttons “*Generate...*” und “*Close*”. Nachdem das gewünschte Projekt aus der Liste ausgewählt wurde, kann der “*Generate*”-Button gedrückt werden. Daraufhin öffnet sich ein Dialog-Fenster zur Eingabe des Zielverzeichnisses für die Berichtdateien. Mit *OK* wird die Eingabe bestätigt und die ausgewählten Dateien werden vom Programm generiert. “*Close*” beendet das Programm.

Die durch die Desktop-Applikation generierten Dokumente und Tabellen dienen dem Projektleiter zur Ergebnisinterpretation (**Anwendungsfall 7**). Durch einen Vergleich mit Ergebnissen älterer Projekte, kann er über den zu erwartenden Verbrauch an Ressourcen (z.B.: Zeit und Geld) Aussagen treffen und er kann das aktuelle Projekt bewerten.

## 5.5 Die Zustände des Function-Point Zähltools

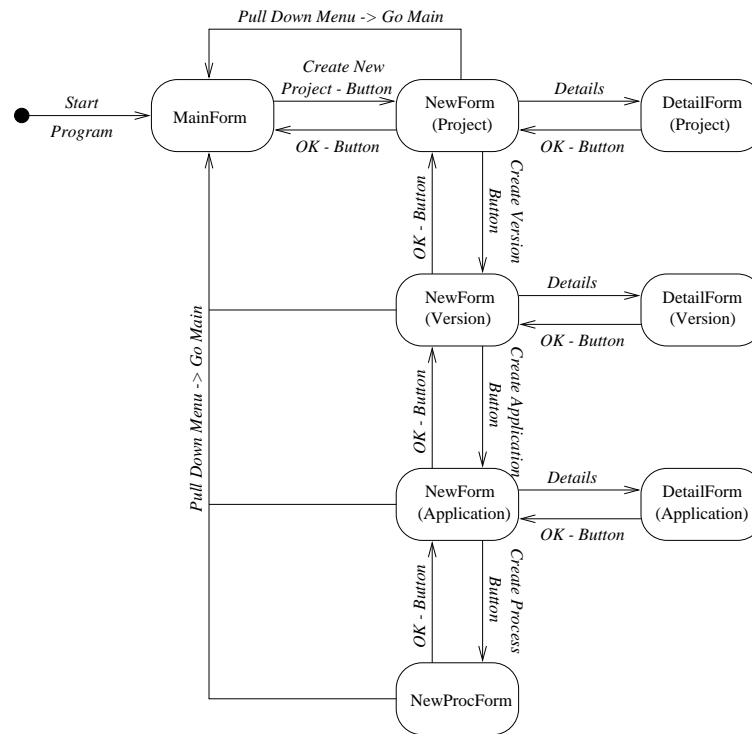


Abbildung 41: Zustandsdiagramm zur Erstellung eines neuen Projektes

Die Abbildungen 41–45 zeigen am Beispiel des User Interfaces die Zustände

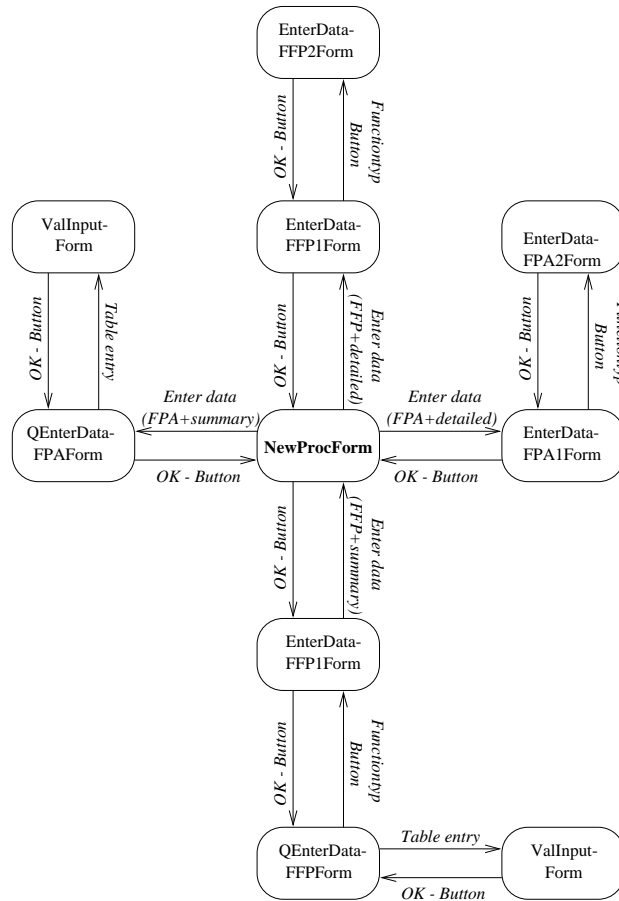


Abbildung 42: Zustandsdiagramm zur Eingabe der Funktionstypen

(hier die Bildschirme) der Palm- und der Desktop-Applikation. Dabei orientieren sie sich an der Szenariobeschreibung (Abschnitt 5.4) des Function-Point Zähltools. Die Zustandsübergänge werden durch Ereignisse mit UI-Elementen in den Bildschirmen ausgelöst.

Abbildung 41 zeigt die Erstellung eines neuen Projektdatensatzes und einer beliebigen Anzahl an zugehörigen Phasen-, Applikations- und Prozessdatensätzen. Der Start des Programms ist hier Ausgangspunkt, von wo der Nutzer mit dem Button *“Create New Projekt”* zu den zugehörigen Bildschirmen gelangt. Schlusspunkt ist in diesem Fall die *NewProcForm*, welche die Optionen bietet, entweder zum Hauptbildschirm per Pull-Down-Menü zurückzukehren, oder aber zur Eingabe der Anzahl der Funktionstypen weiterzugehen (siehe Abbildung 42).



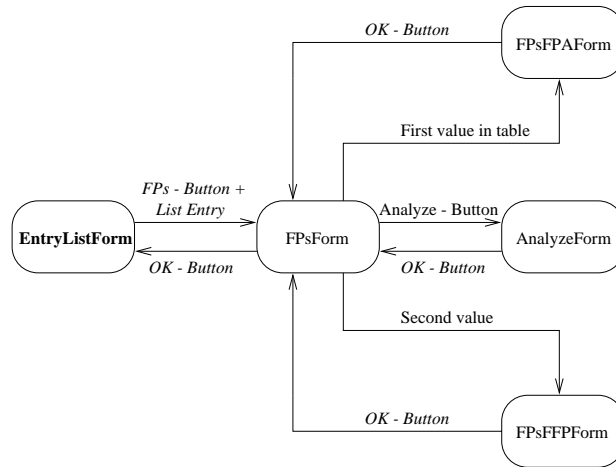


Abbildung 44: Zustandsdiagramm zur Auswertung von Datensätzen

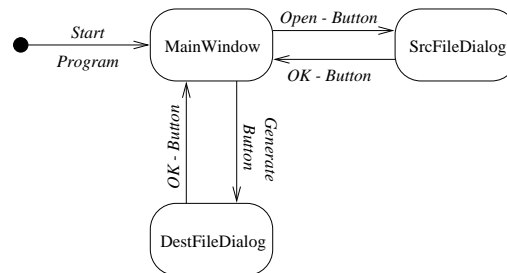


Abbildung 45: Zustandsdiagramm von FPCAnalyze

stellten Möglichkeiten.

Das letzte Diagramm (Abbildung 45) dient zur Darstellung des User Interfaces der Desktop–Applikation "FPCAnalyze".

## 5.6 Die Komponenten von FPC

Wie schon in Abschnitt 2.5 beschrieben, sind Palm–Applikationen ereignisgesteuert. Es existiert in FPC für den Start und das Ende des Programms und für jede **Form** jeweils ein **EventHandler**, wobei die letzteren mit mehreren Ereignissen umgehen können müssen. Folgende Forms gehören zu FPC:

**MainForm:** Der Startbildschirm von FPC.

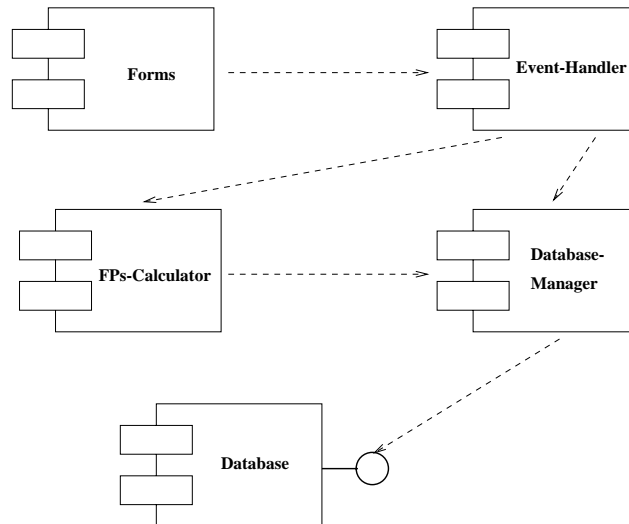


Abbildung 46: Die Komponenten des Function-Point Zähltools

**NewForm:** Bildschirm zur Erstellung eines neuen Projekt-, Phasen- oder Applikationsdatensatzes, mit Eingabefeldern für den Namen und einer Kurzbeschreibung.

**DetailForm:** Bildschirm zur Eingabe weiterer Datensatzdetails, wie Beginn, Ende und Kosten.

**NewProcForm:** Bildschirm zur Erstellung eines neuen Prozessdatensatzes.

**EnterDataFPA1Form:** Bildschirm zur Auswahl der Management-Funktionstypen bei der detaillierten Eingabe der Daten zur Function-Points Berechnung.

**EnterDataFPA2Form:** Komplexitätsmatrix zur detaillierten Eingabe der Zahlen der Management-Funktionstypen.

**EnterDataFFP1Form:**

Bildschirm zur Auswahl der Kontroll-Funktionstypen bei der detaillierten Eingabe der Daten zur Function-Points Berechnung.

**EnterDataFFP2Form:** Komplexitätsmatrix zur detaillierten Eingabe der Zahlen der Kontroll-Funktionstypen.

**QEnterDataFPAForm:** Wertetabelle zur zusammengefassten Eingabe der Zahlen der Management-Funktionstypen.

**QEnterDataFFPForm:** Wertetabelle zur zusammengefassten Eingabe der Zahlen der Kontroll-Funktionstypen.

**ValInputForm:** Universeller Bildschirm zur Eingabe eines einzelnen Zahlenwertes.

**EntryListForm:** Navigationsbildschirm mit Liste für die Namen der Projekte, Phasen, Applikationen und Prozesse.

**FPsForm:** Bildschirm zur Anzeige der berechneten Funktion-Points.

**FPsFPAForm:** Bildschirm zur Anzeige der berechneten Function-Points, zugeordnet zu den einzelnen Management-Funktionstypen.

**FPsFFPForm:** Bildschirm zur Anzeige der berechneten Function-Points, zugeordnet zu den einzelnen Kontroll-Funktionstypen.

**AnalyzeForm:** Bildschirm zur Präsentation der berechneten statistischen Ergebnisse.

Werden in einem der Bildschirme Informationen aus der Datenbank benötigt oder modifiziert, dann benutzt der zugehörige Eventhandler die Funktionen des **Database-Manager**. Beispiele dafür sind das Anlegen eines neuen Projektes, die Abspeicherung der Anzahl von Funktionstypen, das Löschen einer Phase aus einem Projekt oder das Anfordern von Projektdaten zur Berechnung der Function-Points und der statistischen Ergebnisse durch den **FPs-Calculator**. Für den zuletzt genannten Fall müssen beispielsweise alle zum Projekt gehörenden Phasen, Applikationen und Prozesse in der **Datenbank** durchgearbeitet werden, da aus Speicherplatzgründen jeweils nur ein Datensatz von jedem Typ im Arbeitsspeicher gehalten wird.

Der Database-Manager beinhaltet Funktionen zur Erstellung der Datenbank und zum Auffinden, Holen, Lesen, Modifizieren und Löschen von einzelnen Datenbankeinträgen (Records).

In Abbildung 46 sind die Komponenten von FPC und ihre Beziehungen untereinander, noch einmal grafisch dargestellt.

## 5.7 Die Klassen von FPCAnalyze

FPCAnalyze wurde im Gegensatz zu FPC, objektorientiert entwickelt und besteht aus folgenden Klassen (siehe Abbildung 47):

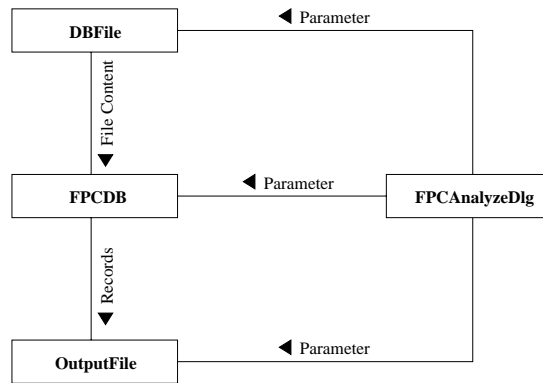


Abbildung 47: Das Klassendiagramm von FPCAnalyze

**FPCAnalyzeDlg** Die Klasse FPCAnalyzeDlg hat die Aufgabe das Userinterface zu initialisieren und auf Eingaben vom Nutzer zu reagieren. Dazu gehört u.a. auch, eine Projektliste zu füllen, die Beschreibung markierter Projekte anzuzeigen, Checklisten zu verwalten und dem Nutzer ein Filedialog zum Auffinden und Auswählen der auf dem PC gesicherten Datenbank anzubieten. Des weiteren versorgt sie andere Klassen mit Parametern, wie z.B. dem Pfad der Datenbankdatei oder der vom Nutzer getroffenen Wahl, welche Berichtdateien erstellt werden sollen.

**DBFile** Methoden dieser Klasse dienen zum Öffnen der Datenbankdatei und zum Extrahieren des Fileheaders mit allgemeinen Informationen (z.B.: Anzahl der gespeicherten Einträge), der Recordliste mit Informationen über die einzelnen Datenbankeinträge (z.B.: deren Größe in Bytes) und der eigentlichen Datenbank. Diese Informationen werden dann zur weiteren Verarbeitung bereitgestellt.

**FPCDB** Die Aufgabe dieser Klasse ist es, die Einträge aus der von DBFile bereitgestellten Datenbank zu entnehmen und in einer eigenen Datenstruktur zu speichern. Somit besteht die Möglichkeit, direkt auf einzelne Einträge über ihren Index oder ihre ID zugreifen zu können. Aufgebaut ist die Datenstruktur aus jeweils einem Array aus Strings für die Projekte, Phasen, Applikationen und Prozesse. Jeder String enthält die Daten von genau einem Datenbankeintrag.

**OutputFile** Die Methoden der letzten Klasse generieren, aus den Informationen der Records aus der Datenbank, die gewünschten Berichte und Tabellen in einem vom Nutzer auszuwählendem Verzeichnis.

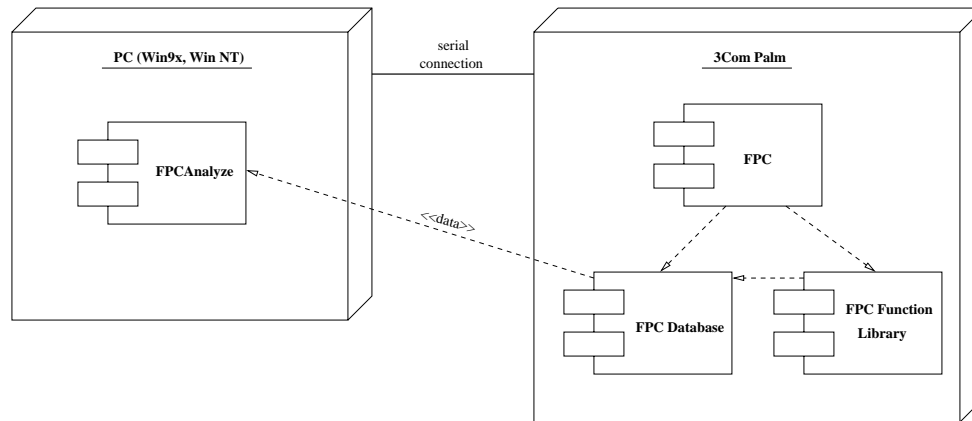


Abbildung 48: Das Verteilungsdiagramm des Function-Point Zähltools

## 5.8 Die Verteilung des Function-Point Zähltools

Wie schon in vorhergehenden Kapiteln erwähnt, besteht das Function-Point Zähltool aus der auf Handheldcomputern mit PalmOS lauffähigen Applikation FPC und der Shared Library FPCLib. Zur Auswertung der FPC-Datenbank und Berichtgenerierung dient dagegen das für MS-Windows entwickelte FPCAnalyze. Die FPC-Datenbank wird beim Hotsync-Vorgang automatisch auf dem PC gesichert. Dazu muss allerdings eine serielle Verbindung mit einem Kabel oder über Infrarot, zwischen Palm und PC, aufgebaut werden. Zur Auswertung und Berichtgenerierung ist diese Verbindung dagegen nicht mehr nötig.

## 5.9 Analyse des Quellcodes mit PC-Metrik

Die Analyse des Quellcodes von FPC, mit dem Software-Messtool PC-Metrik (siehe [DFKW96]), ergab folgende Resultate:

```
PC-METRIC (C) Version 4.0.2
Summary Complexity Report for: FPC.RP2
-----
```

Software Science Length (N):	13100
Estimated Software Science Length (N <sup>^</sup> ):	5199
Software Science Volume (V):	121148

Software Science Effort (E):	71822037
Estimated Errors using Software Science ( $B^{\wedge}$ ):	38
Estimated Time to Develop, in hours ( $T^{\wedge}$ ):	1108
Cyclomatic Complexity (VG1):	384
Extended Cyclomatic Complexity (VG2):	396
Average Cyclomatic Complexity:	12
Average Extended Cyclomatic Complexity:	12
Average of Nesting Depth:	3
Average of Average Nesting Depth:	1
Lines of Code (LOC):	3011
Physical Source Stmts (PSS):	2889
Logical Source Stmts (LSS):	1921
Nonexecutable Statements:	75
Compiler Directives:	71
Number of Comment Lines:	375
Number of Comment Words:	1392
Number of Blank Lines:	122
Number of Procedures/Functions:	31

## 5.10 Fazit und Ausblick

Die Entwicklung und Implementierung des Function-Point Counters kann als abgeschlossen betrachtet werden. Die in den Anforderungen aufgeführten Features wurden umgesetzt, alle bekannten Bugs beseitigt und eine Dokumentation geschrieben. Beim Entwurf und bei der Umsetzung sind aber neue Ideen zur Verbesserung und Weiterentwicklung entstanden. Gerade bei der Desktop-Applikation ist dazu noch viel Potenzial vorhanden. Denkbar wäre z.B. ein Programm, welches die Eingabe von neuen und die Modifizierung von bestehenden Daten auf dem PC ermöglicht und aus diesen Informationen eine FPC-Palmdatenbank erzeugt. Dadurch würde wiederum ein Conduit notwendig, welches die Synchronisation der Datenbanken auf dem Palm und dem PC sicherstellt.

Eine weitere Verbesserung der Desktop-Applikation ist die Nutzung des RTF-Formates bei der Berichtgenerierung. Dies würde eine Vorformatierung der Ausgabedokumente ermöglichen und dem Nutzer einiges an Arbeit, beim

Import der Dokumente in gängige Textverarbeitungsprogramme, ersparen. Eine weitere Möglichkeit ist die Portierung der Desktop–Applikation auf UNIX–Betriebssysteme, wie Linux oder Solaris, wo schon Programme zur Synchronisation mit dem Palm existieren (z.B.: Pilot-Link<sup>31</sup>). Damit kann eine größere Plattformunabhängigkeit erlangt werden. Allerdings wäre dies wohl am besten zu erreichen, wenn FPCAnalyze komplett in der Programmiersprache JAVA implementiert würde.

Aber auch die Palm–Applikation bietet noch einen weiten Raum an Verbesserungsmöglichkeiten. Gerade was die Kommunikation und der Datenaustausch zwischen zwei Handheldcomputern betrifft. Sollten beispielsweise zwei oder mehr Counter an einem Zählprojekt beteiligt sein, so sollte die Möglichkeit bestehen, die Ergebnisse der Analyse auszutauschen und die Datenbanken zu synchronisieren, um damit ein verteiltes Arbeiten zu ermöglichen.

Als letztes ist in diesem Zusammenhang noch die Nutzung anderer programmierbarer Handheldcomputer und Tastatur–Organizer zu erwähnen. Beispielsweise die Psion–Plattform und Windows CE–Computer haben neben den Palm eine relativ hohe Verbreitung gefunden. Den Function–Point Counter auch auf diese Plattformen zu portieren, würde die Anzahl potentieller Nutzer wesentlich vergrößern. Allerdings ist dies mit einem sehr hohen Aufwand verbunden, da sich die Architektur dieser Systeme sehr stark von der des Palms unterscheidet.

---

<sup>31</sup><http://www.pilot.pasta.cs.uit.no/index.html>

## 6 Zusammenfassung

### 6.1 Ergebnisse

Das Ergebnis der vorliegenden Diplomarbeit ist die Beschreibung von Möglichkeiten für den Einsatz des Palm-Handheldcomputers zur Unterstützung des Software-Entwicklungsprozesses.

Aufgrund der Analyse der System bedingten Vor- und Nachteile des Palms sowie der Untersuchung des Softwarelebenszyklus auf verwendete Methoden und unterstützende Werkzeuge, entstanden Ideen für die Nutzung bestehender Palm-Applikation (z.B.: Datenbank-Anwendungen, Notizzettel-Programme, Aufgaben-Verwaltungen), und Lösungen für die Umsetzung von CASE-Tools auf den Palm (z.B.: Messtool, Projektverwaltungs-Programme, Qualitätsmanagement-Programme). Allerdings wurde auch ziemlich schnell klar, für welche Einsatzzwecke sich der Palm nicht eignet. Immer wenn mit umfangreichen Datenmengen gearbeitet werden muss oder viel Speicherplatz, eine große Anzeige oder einen schnellen Prozessor benötigt wird, ist die Nutzung des Palms kaum oder nur mit Einschränkungen möglich. Das betrifft z.B. die Modellierung von Softwarekomponenten, die Implementierung mit integrierten Entwicklungsumgebungen und das Niederschreiben der Dokumentation mit Textverarbeitungsprogrammen.

Begründet durch die zeitlichen Beschränkungen einer Diplomarbeit, konnte der Entwurf der einzelnen Lösungsvorschläge nicht detailliert genug vorgenommen werden.

Die Betrachtung des sehr großen Software-Angebotes für den Palm-Computer ergab, dass zur Zeit, bis auf ein paar Projektmanagement-Programme, kaum Lösungen auf diesem Gebiet existieren. Einzig die Recherche über das Internet wird mit einer Anzahl von HTML- und WAP-Browsern unterstützt.

### 6.2 Ausblick

Der nächste logische Schritt wäre die Realisierung der in dieser Arbeit vorgestellten Lösungen. Dabei sollte aber erst einmal im Vorfeld überprüft werden, ob es sich nicht vielleicht lohnt, die Werkzeuge in einer CASE-Plattform zu integrieren. Diese CASE-Plattform würde die Tools unter einer einheitlichen Nutzeroberfläche zusammenfassen, Funktionen für den Zugriff auf die Datenbank anbieten und Schnittstellen zur Integration neuer Tools definieren.

Aber auch das Function-Point Zähltool (FPC) könnte noch verbessert und erweitert werden:

- Implementierung eines vollwertigen Conduits zur Synchronisierung des Datenbestandes mit dem PC.
- Erweiterung der Desktop-Applikation mit Möglichkeiten zur Verwaltung der Projektdateibanken und Eingabe neuer Daten.
- Portierung des Function-Point Zähltools auf andere Handheldcomputer, wie z.B. Windows CE- und EPOC-Plattformen.
- Portierung der Desktop-Applikation auf andere Betriebssysteme, wie z.B. Linux oder MacOS.
- Ermöglichung der verteilten Bearbeitung von Zählprojekten mit mehreren Palm-Computern.
- Erhöhung der Effizienz beim Datenbankzugriff.
- Einbettung des Programms in die zu realisierende CASE-Plattform.
- Beobachtung der Weiterentwicklung der FFP-Methode und wenn nötig, Anpassung von FPC.

Des Weiteren sollten auch die Entwicklungen auf dem Gebiet der Handheldcomputer weiter beobachtet werden. Schnellere Prozessoren, mehr Speicher und neue Technologien könnten eine Neubetrachtung des Softwarelebenszyklusses, im Zusammenhang mit den genutzten CASE-Tools, notwendig machen. Dabei müssen auch die neuen Erkenntnisse im Bereich der Softwaretechnik und speziell, des computergestützten Software-Engineering, mit einbezogen werden.

# A Endnutzerdokumentation des Function–Point Zähltools

## A.1 Einleitung

Der Function–Point–Counter (kurz: FPC) ist ein palmbasiertes Tool zur Unterstützung des Function–Point Zählprozesses. Mit ihm können die ermittelten Daten komfortabel elektronisch erfasst, verwaltet, ausgewertet und aufbereitet werden. Eine Palm–Applikation übernimmt die Eingabe, die Verwaltung und eine erste Berechnungen der Daten und die abschließende Auswertung und Berichtgenerierung erfolgt unter Windows. Durch die Nutzung eines Handheldcomputers wird eine mobile Erfassung und Präsentation der Daten ermöglicht, damit ist der Counter beim Zählprozess, von Desktop–Computern weitestgehend unabhängig.

Benötigt wird ein Handheld–Computer mit PalmOS ab der Version 2 mit ca. 100 KB freiem Speicher und ein PC mit Windows 9x, Windows NT oder höher. Damit auf die Daten unter Windows zugegriffen werden kann, muss ein Programm zur Daten–Synchronisation mit dem Palm (z.B. Palm Desktop) existieren.

Das Produkt besteht aus folgenden Dateien:

**fpc.prc:** Palm–Applikation,

**FPCLib.prc:** Bibliothek, die von der Palm–Applikation benötigt wird,

**FPCAnalyze.exe:** Windows–Applikation,

**fpc.pdf:** Dokumentation zum Function–Point–Counter

### A.1.1 Funktionsweise

FPC unterstützt die Berechnung von Function–Points nach der Methode der Full–Function–Points (FFP) und damit auch automatisch nach der Methode der Function–Point–Analyse (FPA). Dabei werden von FPC pro "Prozess" nur die Zahlen der verschiedenen Funktionstypen abgespeichert. Die Berechnung und Auswertung der Function–Points erfolgt durch das Programm. Die Namen der verschiedenen Funktionstypen sind in den beiden folgenden Tabellen aufgeführt.

<b>Management–Funktionstypen</b>
Internal Logical File (ILF)
External Interface File (EIF)
External Input (EI)
External Output (EO)
External Inquiry (EQ)
<b>Kontroll–Funktionstypen</b>
single occurrence Updated Control Group (s.o. UCG)
single occurrence Read-only Control Group (s.o. RCG)
multiple occurrence Updated Control Group (m.o. UCG)
multiple occurrence Read-only Control Group (m.o. RCG)
External Control Entry (ECE)
External Control Exit (ECX)
Internal Control Read (ICR)
Internal Control Write (ICW)

In FPC ist ein "Prozess" ein funktioneller Teil einer komplexen "Applikation" und entspricht der "counting boundary" der Function–Point Zählmethode. Eine Applikation besteht aus einer bestimmten Anzahl von Prozessen und gehört in FPC immer zu einem "Softwareprojekt". Dadurch wird es ermöglicht, auf einem Palm mehrere verschiedene Software–Projekte gleichzeitig zu verwalten und zu vergleichen. In seiner Entstehung durchläuft ein Software–Produkt mehrere Phasen, wobei verschiedene Dokumente den jeweils aktuellen Entwicklungsstand präsentieren (z.B.: Entwurfsdokumente, Spezifikationen, Beschreibungen des fertigen Produktes oder von verschiedenen Produktversionen). Diese Dokumente werden häufig "gezählt", um den Entstehungsprozess besser verfolgen zu können. In FPC entspricht die "Version" den verschiedenen Phasen oder Produktversionen des Softwareprojektes. Damit ergibt sich die folgende Datenstruktur der FPC–Datenbank (siehe auch Abbildung 31, auf Seite 71):

- Die FPC–Datenbank besteht aus einer Anzahl von Softwareprojekten.
- Ein Softwareprojekt besteht aus einer Anzahl von Versionen.
- Eine Version besteht aus einer Anzahl von Applikationen.
- Eine Applikation besteht aus einer Anzahl von Prozessen.

### A.1.2 Die Elemente der verschiedenen Datensätze

Folgende Informationen können für jedes Softwareprojekt eingegeben werden:

- Name des Projektes,
- eine kurze Beschreibung des Projektes,
- Beginn des Projektes,
- Ende des Projektes (wenn bekannt),
- die Gesamtkosten für das Projektes (wenn bekannt).

Folgende Informationen können für jede Version eingegeben werden:

- Name der Version,
- eine kurze Beschreibung der Version,
- Beginn der Version,
- Ende der Version (wenn bekannt),
- die Gesamtkosten für die Version (wenn bekannt).

Folgende Informationen können für jede Applikation eingegeben werden:

- Name der Applikation,
- eine kurze Beschreibung der Applikation,
- Beginn der Applikation,
- Ende der Applikation (wenn bekannt),
- die Gesamtkosten für die Applikation (wenn bekannt),
- der Name des Counters,
- der Name des gezählten Dokumentes.

Folgende Informationen können für jeden Prozess eingegeben werden:

- Name des Prozesses,
- die Nummer, des zum Prozess gehörenden Abschnittes, im gezählten Dokument,
- die Anzahl der Seiten der Prozessbeschreibung,
- die Zahlen der verschiedenen Management–Funktionstypen,
- die Zahlen der verschiedenen Kontroll–Funktionstypen.

## A.2 Die Bedienung der Palm Applikation

### A.2.1 Erstellung eines neuen Projektes

Über den Button "*Create New Project*" im Startbildschirm der Palm–Applikation gelangt man zu den Bildschirmen zur Erstellung eines neuen Projektdatensatzes (siehe linker Bildschirm in Abbildung 34, auf Seite 76). Hier können der Name und die Kurzbeschreibung des Softwareprojektes sowie unter "*Details*", die Informationen über den zugehörigen Zeitraum und den Kosten eingegeben werden (rechter Bildschirm in Abbildung 34, auf Seite 76). Das Datumsformat für den Beginn– und Endzeitpunkt ist "*Monat / Tag / Jahr*" und es werden nur vierstellige Jahreszahlen akzeptiert. Menüpunkte bieten an, den Datensatz zu speichern oder zum Startbildschirm zurückzukehren. Wurden alle gewünschten Daten eingegeben, so gelangt man durch den "*OK*"–Button wieder zum Startbildschirm oder durch den "*Create Version*"–Button, zu den Bildschirmen zur Erstellung eines neuen Versionsdatensatzes. In beiden Fällen werden die eingegebenen Informationen der FPC–Datenbank hinzugefügt. Da der Speicherplatz des Palm beschränkt ist, sollten Name und Beschreibung so kurz wie möglich sein. In die FPC–Datenbank passen ca. 600 Datensätze.

Die Bildschirme zur Erstellung eines neuen Versionsdatensatzes entsprechen weitestgehend den Bildschirmen zur Erstellung eines neuen Zählprojektes. Wurden alle gewünschten Daten eingegeben, so gelangt man durch den "*OK*"–Button zurück zum aktuellen Projektdatensatz und kann eine weitere Version erstellen. Mit "*Create Application*" gelangt man zu den Bildschirmen zur Erstellung eines neuen Applikationsdatensatzes.

Auch in diesen Bildschirm kann der Name, eine Beschreibung, der Beginn und das Ende sowie die Kosten eingegeben werden. Zusätzlich dazu existieren unter "*Details*", Eingabemöglichkeiten für den Name des Counters und den

Namen des zu zählenden Dokumentes. "OK" und "Create Process" schließen, nach Speicherung der neuen Informationen, die Bildschirme zur Erstellung eines neuen Applikationsdatensatzes und man gelangt entweder zurück zum aktuellen Versionsdatensatz oder zum Bildschirm zur Erstellung eines neuen Prozessdatensatzes.

Der Bildschirm zur Erstellung eines Prozessdatensatzes unterscheidet sich

Abbildung 49: Bildschirm zur Erstellung eines neuen Prozessdatensatzes

größtenteils von den vorhergegangenen (siehe Abbildung 49). Es existiert ein Eingabefeld für den Namen des Prozesses, eins für die Abschnittsnummer im gezählten Dokument ("Sectionnumber:") und eins für die Anzahl der Seiten, die den Prozess beschreiben (für statistische Berechnungen). Mit "Counting Method:" und "Inputtype:" können für die folgende Eingabe der Daten zu den Funktionstypen, noch Einstellungen getroffen werden. Die erste Wahlmöglichkeit entscheidet über die Art der Funktionstypen zu denen Daten eingegeben werden sollen, damit sind die Management Funktionstypen der FPA-Methode und die Kontroll Funktionstypen der FFP-Methode gemeint und die zweite Wahlmöglichkeit entscheidet über die Art der Eingabe der Daten (siehe Abschnitt A.2.2).

Mit dem Button "Enter Data" gelangt man zur Eingabe der Zahlen zu den verschiedenen Funktionstypen und "Done" schließt den Bildschirm, speichert den Prozess in der Datenbank und führt zum aktuellen Applikationsdatensatz zurück.

### A.2.2 Eingabe der Daten

Es gibt zwei Möglichkeiten Daten zu den verschiedenen Funktionstypen einzugeben: Detailliert ("detailed") oder Zusammengefasst ("summary"). Im ersten Fall wird davon ausgegangen, dass der Counter eine Prozessbeschreibung

Schritt für Schritt durchgeht und die erlangten Informationen ohne Zwischenspeicherung sofort auf den Palm übertragen will. Die zweite Möglichkeit setzt voraus, dass alle Funktionstypen eines Prozesses schon komplett erfasst und aufgeschrieben wurden. Der Counter braucht sie dann bloß noch zu übertragen.

Mit der zusätzlichen Auswahl der Art der Funktionstypen ergeben sich vier Möglichkeiten der Eingabe:

1. Detaillierte Eingabe der Management Funktionstypen,
2. Zusammengefasste Eingabe der Management Funktionstypen,
3. Detaillierte Eingabe der Kontroll Funktionstypen,
4. Zusammengefasste Eingabe der Kontroll Funktionstypen.

Wurde die erste Möglichkeit gewählt ("FPA", "detailed"), so erscheint ein Bildschirm mit den fünf verschiedenen Management Funktionstypen, jeweils gefolgt von einem Button. Nach Drücken eines der Buttons gelangt man zu einer entsprechenden Komplexitätsmatrix (siehe linker Bildschirm in Abbildung 36, auf Seite 78), wo die Komplexität der Funktion eingestellt werden kann (vertikal die Anzahl der RETs, horizontal die Anzahl der DETs). Mit "OK" wird die Wahl bestätigt und mit "Cancel" verworfen. In beiden Fällen gelangt man zum vorherigen Bildschirm zurück und es kann mit der nächsten Funktion fortgefahren werden. Die Zuordnung der Funktion zu den Komplexitätsgraden "low", "average" und "high" geschieht automatisch. "Done" beendet die Eingabe von Werten und führt zum Prozessbildschirm zurück.

Die Wahl der zweiten Möglichkeit ("FPA", "summary") öffnet einen Bildschirm mit einer Tabelle für die Zahlen der verschiedenen Funktionstypen, zugeordnet zu den entsprechenden Komplexitätsgraden "low", "average" und "high" (rechter Bildschirm in Abbildung 36, auf Seite 78). Nach der Berührung eines der Tabelleneinträge mit dem Stift, kann in einem Eingabefeld ein neuer Wert eingegeben werden, der mit "OK" bestätigt werden muss. "Cancel" verwirft die neue Eingabe. Wurden alle Werte in der Tabelle eingegeben, so gelangt man durch "OK" zum Prozessbildschirm zurück.

Sollen Daten für die Kontroll Funktionstypen in die FPC-Datenbank übertragen werden ("FFP"), so erscheint in den Fällen drei und vier ein Bildschirm zur Auswahl des entsprechenden Funktionstyps.

Die Buttons führen, in Abhängigkeit der Eingabemethode ("detailed" oder

”summary”), zu weiteren Eingabebildschirmen. Bei den Data Function Types ”Updated Control Data (UCG)” und ”Read-only Control Data (RCG)” steht der ”s”-Button für eine ”single occurrence group of data” und der ”m”-Button für eine ”multiple occurrence group of data”. Hier eine Unterscheidung vorzunehmen ist notwendig, da jeder der beiden Funktionstypen eine andere Eingabemaske erfordert.

Die beiden Screenshots in Abbildung 50 zeigen Bildschirmmasken zur detail-

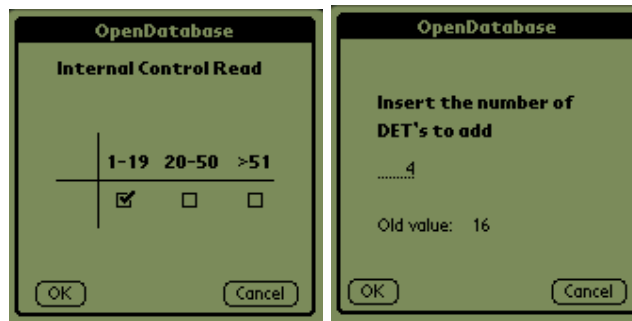


Abbildung 50: Bildschirme zur detaillierten Eingabe der Kontroll Funktionstypen

lierten Eingabe der Kontroll-Funktioinstypen (”FFP”, ”detailed”). Der Bildschirm ganz links dient zur Eingabe der ”Control Transactional Function Types”. Die Zuordnung der Funktion ist abhängig von der Anzahl der DETs. Im rechten Bildschirm wird die Anzahl der DETs der ”single occurrence groups of data” eingegeben, da laut Definition pro Applikation nur maximal eine ”Update Control Group” und eine ”Read-only Control Group” existieren kann. Dieser Wert wird dem alten Wert nach Bestätigung des ”OK”-Buttons hinzuaddiert.

Der Bildschirm zur detaillierten Eingabe der ”multiple occurrence groups of data”, entspricht weitestgehend dem Bildschirm zur detaillierten Eingabe der ”Management Data Function Types” ”ILF” und ”EIF”. Die Auswahl erfolgt hier analog.

Bei der zusammengefasste Eingabe der Werte zu den ”Control Transactional Function Types” und den ”multiple occurrence groups of data” (”FFP”, ”summary”), werden in der dargestellten Tabelle alle bisherigen Werte zu der entsprechenden Funktion angezeigt. Soll einer der Werte geändert werden, so ist dieser anzuwählen. Daraufhin erscheint ein Dialog, in dem der neue Wert eingegeben werden kann. Die Tabelle wird danach aktualisiert.

### A.2.3 Modifizierung von Datensätzen

Es gibt verschiedenen Gründe die es notwendig machen, in FPC Datensätze zu modifizieren:

- Daten und Zählergebnisse müssen korrigiert werden,
- neue Zählergebnisse müssen hinzugefügt werden,
- neuen Versions-, Applikations- oder Prozessdatensätzen eines Projektes müssen erstellt werden,
- existierende Datensätze müssen gelöscht werden.

Um einzelne Datensätze gezielt modifizieren zu können, muss eine Möglichkeiten existieren, durch die FPC-Datenbank zu "browsen". Nach Wahl des "*Modify Projekt*"-Buttons gelangt man zu den in Abbildung 37, auf Seite 79 dargestellten Navigationsbildschirmen. Der linke Bildschirm besteht aus einer Liste mit allen vorhandenen Projekten, einer Buttonreihe und einem "*OK*" Button. Hier hat man nun mehr Möglichkeiten:

1. "*Del*" und die Wahl eines Eintrages in der Liste, löscht nach einer Sicherheitsabfrage das entsprechende Projekt, mit all seinen Versions-, Applikations- und Prozessdatensätzen in der FPC-Datenbank.
2. "*Info*" und die Wahl eines Eintrages in der Liste, ermöglicht die Modifizierung des Projektdatensatzes und die Generierung weiterer Versions-, Applikations- und Prozessdatensätze, durch die Anzeige der aktuellen Projektdaten in den schon bekannten Bildschirmen aus Abschnitt "Erstellung eines neuen Projektes". Mit dem Button "*Create Version*" kann auch hier ein neuer Versionsdatensatz erstellt werden und mit "*OK*" gelangt man zurück zur Übersicht.
3. Durch "*Ver's*" und die Wahl eines Listeneintrages werden in der Liste alle dem Projekt zugehörigen Versionen angezeigt (siehe rechtes Bild in Abbildung 37, auf Seite 79).
4. Mit dem "*OK*"-Button gelangt man zum Startbildschirm zurück.

Die dritte Möglichkeit dient zur Navigation durch die Struktur der FPC-Datenbank. Von den Projektdatensätzen gelangt man so zu den Versionsdatensätzen des ausgewählten Projektes. Jetzt hat man die Möglichkeit einen Versionsdatensatz inklusive aller zugehörigen Applikations- und Prozessdatensätze zu löschen oder man modifiziert die Daten einer bestimmten Version und kann dann auch weitere Applikationen und Prozesse erstellen. Als dritte Möglichkeit kann man sich mit dem Button "App's" alle zu einer Version gehörenden Applikationen anzeigen lassen. Mit dem "OK"-Button gelangt man wieder zur Projektübersicht. So kann man mit dem Navigationsbildschirm auf alle Datensätze gezielt zugreifen. Mit den Buttons "Ver's", "App's" und "Proc's" gelangt man jeweils zum nächsten Datensatztyp und mit "OK" wieder zurück.

Wenn man schnell wieder zum Startbildschirm gelangen will, kann man zusätzlich auch den Menüpunkt "Options  $\Rightarrow$  Go Main" auswählen.

#### A.2.4 Auswertung

Zu den Bildschirmen, zur Auswertung der eingegebenen Daten, gelangt man durch den Button "Analyze Project". Hier erscheint der selbe Navigationsbildschirm wie in Abschnitt A.2.3, mit dem einen Unterschied, dass der "Del"-Button durch den "FPs" Button zur Analyse ersetzt wurde (siehe Abbildung 38, auf Seite 80). Das hat den Vorteil, dass wenn gewünscht, alle Datensatztypen auch einzeln ausgewertet werden können.

**Function-Points** Wurde der "FPs" Button ausgewählt und danach ein Datensatz aus der Liste, dann werden die Zahlen der verschiedenen Funktionstypen zusammengezählt, die Function-Points berechnet und es erscheint der in Abbildung 39, auf Seite 80 links dargestellte Bildschirm. In der ersten Zeile stehen die Function-Points der Management-Funktionstypen, in der zweiten Zeile die Function-Points der Kontroll-Funktionstypen und in der letzten die Anzahl der Unadjusted Function-Points ( $UAF = FPA + FFP$ ). Wird einer der beiden oberen Werte angeklickt, so werden die Function-Points aufgeschlüsselt zu den zugehörigen Funktionstypen angezeigt.

**Statistische Berechnungen** Mit dem "Analyze"-Button im oben dargestellten Bildschirm kann man sich noch weitere statistische Werte anzeigen

lassen (rechter Bildschirm in Abbildung 39, auf Seite 80). Das sind im folgenden:

- Die Anzahl der gezählten Seiten (Pages),
- Die Anzahl der zugehörigen Prozesse (Processes),
- Die durchschnittliche Anzahl von Seiten pro Prozess (Pages/Proc),
- Die durchschnittliche Anzahl von Function-Points pro Seite (FPs/Page),
- Die durchschnittlichen Kosten pro Function-Point (Costs/FP). Dieser Wert ist wichtig um die Kosten für nachfolgende Projekte abschätzen zu können.
- Die Dauer des Projektes, der Version oder der Applikation in Tagen.

### A.3 Die Bedienung der Windows Applikation

Die Windowsapplikation "*FPCAnalyze.exe*" dient der Auswertung von Zählprojekten und der Berichtgenerierung. Dabei wird eine FPC-Datenbank, welche durch den Hotsync-Vorgang automatisch auf dem PC gesichert wird, eingelesen und nach vorhandenen Zählprojekten durchsucht. Nach Auswahl eines Projektes können die folgenden drei Dateitypen aus den Daten des Projektes erzeugt werden:

1. Die erste Datei ("*Projektname.dat*") enthält die eingegebenen Informationen aller Datensätze des Zählprojektes. Das schließt die Zahlen der Funktionstypen bei den Prozessen mit ein.
2. Die zweite Datei ("*Projektname.sta*") enthält die Function-Points der Funktionstypen und statistische Berechnungen der einzelnen Datensätze.
3. "*Tables*" ist ein Verzeichnis, welches Tabellen mit den Zahlen der verschiedenen Funktionstypen und den ermittelten Function-Points zu den Datensätzen Projekt, Version, und Applikation enthält ("*Projektname.tab*", "*Versionsname.tab*", "*Applikationsname.tab*"). Jede Datei enthält eine Tabelle und trägt den Namen des Projektes, der Version oder der Applikation.

Nach dem Start der Applikation erscheint das in Abbildung 40, auf Seite 82 dargestellte Fenster. Es ist folgendermaßen aufgeteilt: Im oberen Textfeld steht die Position der FPC-Datenbank, welche mit dem "Open"-Button und dem daraufhin erscheinenden Dateidialogfeld ausgewählt wird (gewöhnlich FCDB.pdf im Verzeichnis ... \Palm Desktop \Nutzername \Backup). Den linken mittleren Teil nimmt eine Liste, zur Anzeige der in der Datenbank abgespeicherten Zählprojekte sowie ein Feld mit der Beschreibung des ausgewählten Projektes ein. Auf der rechten Seite des Applikationsfensters kann mit den Checkboxen entschieden werden, welche Berichtdateien erstellt werden sollen und ganz unten befinden sich die Buttons "Generate..." und "Close". Nachdem das gewünschte Projekt aus der Liste ausgewählt wurde, kann der "Generate"-Button gedrückt werden. Daraufhin öffnet sich ein Dialogfenster zur Eingabe des Zielverzeichnisses für die Berichtdateien. Mit "OK" wird die Eingabe bestätigt und die ausgewählten Dateien werden vom Programm generiert. "Close" beendet das Programm.

#### Der Aufbau der ersten und zweiten Datei:

Zählprojekt,
erste Version,
erste Applikation, der ersten Version,
erstes Projekt, der ersten Applikation, der ersten Version,
zweites Projekt, der ersten Applikation, der ersten Version,
drittes Projekt, der ersten Applikation, der ersten Version,
...
zweite Applikation, der ersten Version,
erstes Projekt, der zweiten Applikation, der ersten Version
...
dritte Applikation, der ersten Version
...
zweite Version,
...

**Die Informationen der jeweiligen Datensätze in der ersten Datei:**

<b>Projekt</b>
Name des Projektes, kurze Beschreibung des Projektes, Kosten des Projektes, Anzahl an Versionen
<b>Version</b>
Name der Version, Name des zugehörigen Projektes, kurze Beschreibung der Version, Kosten der Version, Anzahl an Applikationen
<b>Applikation</b>
Name der Applikation, Name der zugehörigen Version, kurze Beschreibung der Applikations, Anzahl an Prozessen, Name des Counters, Name des gezählten Dokuments
<b>Prozess</b>
Name des Prozesses, Name der zugehörigen Applikation, Abschnittsnummer im gezählten Dokument, Anzahl der gezählten Seiten, Zahlen der Management Funktionstypen, Zahlen der Kontroll Funktionstypen

**Die Informationen der jeweiligen Datensätze in der zweiten Datei:**

Typ des Datensatzes,
Name des Datensatzes,
Unadjusted Function-Points,
Anzahl an Versionen (nur bei Projekt),
Anzahl an Applikationen (nur bei Projekt und Version),
Anzahl an Prozessen (nur bei Projekt, Version und Applikation),
Dauer in Tagen,
Kosten,
Anzahl an gezählten Seiten,
Durchschnittlichen Kosten pro Function-Point,
Durchschnittliche Anzahl von Seiten pro Prozess,
Durchschnittliche Anzahl von Function-Points pro Tag,
Durchschnittliche Anzahl von Function-Points pro Prozess,
Durchschnittliche Anzahl von Function-Points pro gezählter Seite,
Function-Points der einzelnen Management Funktionstypen,
Function-Points der einzelnen Kontroll Funktionstypen

**Der Aufbau der Tabellen<sup>32</sup>:**

- In der Projekttable sind alle Versionen, mit den Zahlen der Funktionstypen und ihren Function-Points aufgelistet.
- In den Tabellen der Versionen sind die zugehörigen Applikationen mit den Zahlen der Funktionstypen und ihren Function-Points aufgelistet.
- In den Tabellen der Applikationen sind die zugehörigen Prozesse mit den Zahlen der Funktionstypen und ihren Function-Points aufgelistet.

---

<sup>32</sup>Die Spalten der Tabellen sind durch Tabulator-Zeichen getrennt. Damit können diese Dateien in die gängigen Tabellenkalkulations-Programme importiert werden.



## B Beispiel Function–Point Zählung

Abschnitt B.1 enthält Ausschnitte aus dem Inhalt, der durch das Function–Point Zähltools generierten Ergebnisdokumente, einer Function–Point Zählung. Die erste Datei (*fpc.dat*) enthält die Informationen der Datenbank des gewählten Projektes und die zweite Datei (*fpc.sta*) enthält berechnete Resultate. Datensätze des gleichen Typs sind mit einer wagerechten Linie voneinander getrennt und Datensätze verschiedenen Typs mit zwei wagerechten Linien.

Grundlage für die Zählung ist das Function–Point Zähltool selbst, mit seiner Entwickler– und Endnutzerdokumentation aus Kapitel 5 und Anhang A, sowie seinem Sourcecode.

Folgende Regeln wurden für den Zählprozess festgelegt:

- Der Projektname lautet "*Function–Point Counter*".
- Die Produktversion ist 1.0.
- Gezählt wurde die Palm–Applikation ("application boundary").
- Als funktionelle Prozesse dienen die verschiedenen Bildschirme, mit den dahinterliegenden Funktionen und Datenstrukturen.
- Der Konfigurations–Record und jeder Datenstamm an Projekten, Versionen, Applikationen und Prozessen werden jeweils als ein "External Interface File" gezählt (damit enthält die Datenbank insgesamt 5 EIFs).
- Die aktuell bearbeiteten Projekt–, Versions–, Applikations– und Prozessrecords werden jeweils als ein "Internal Logical File" gezählt (von jedem Typ wird immer nur einer im Programmspeicher gehalten).
- Alle anderen zusammenhängenden Daten entsprechen jeweils einem Internal Logical File.
- Die Betätigung von Kontrollelementen (Buttons, Listen usw.) in einem Bildschirm entspricht einem "External Input", wobei alle Kontrollelemente eines Bildschirms zusammengefasst betrachtet werden.
- Wird auf die Datenbank lesend zugegriffen, handelt es sich um ein "External Inquiry".

- Bei einem schreibenden Zugriff auf die Datenbank handelt es sich um ein External Output.
- Immer wenn Daten eines Records oder Ergebnisse aus Berechnungen auf dem Bildschirm ausgegeben werden, wird ein "External Output" gezählt.
- Die Eingabe von Daten und Werten in Bildschirmschablonen durch den Nutzer entspricht einem "External Input", wobei alle Eingabemöglichkeiten eines Bildschirms zusammengefasst werden.

Die Zählung der Funktionstypen der verschiedenen Prozesse erfolgte mit Hilfe der Komplexitätsmatrix (siehe Abschnitt 5.1) von "FPC".

## B.1 Resultate

### fpc.dat

```

*****
                Function Point Counter
                Project Data File
*****

Projectname      : FPC
Projectdescription : Function-Pointz{"a}hltool

Project costs    : 5000
Number of versions : 1

=====

Type             : Version
Name             : Version 1.0

Project          : FPC

Description      : [none]
Costs            : 5000
Number of Applications : 2

-----

Type             : Application
Name             : FPC

```

Version : Version 1.0  
 Description : Palm-Application  
 Costs : 3500  
 Number of Processes : 23  
 Counter : Daniel Reitz  
 Document : dipl.ps/source

-----  
 Type : Process  
 Name : MainForm

Application : FPC

Section : A.2  
 Number of Pages : 1

Number of functions  
 -----

FPA - Function Types:

	l	a	h	#
ILFs:	2	0	0	2
EIFs:	1	0	0	1
EIs :	1	0	0	1
EOs :	1	0	0	1
EQs :	0	0	0	0

FFP - Function Types:

DETs:	1-19	20-50	>51	#
ICRs:	0	0	0	0
ICWs:	0	0	0	0
ECXs:	0	0	0	0
ECEs:	0	0	0	0

Multiple occurrence groups of data:

	l	a	h	#
UCG:	0	0	0	0
RCG:	0	0	0	0

Single occurrence groups of data:

UCG : 0 DETs  
 RCG : 0 DETs

-----

Type : Process  
 Name : NewProjForm  
 Application : FPC  
 Section : A.2.1  
 Number of Pages : 1

Number of functions  
 -----

FPA - Function Types:

	l	a	h	#
ILFs:	2	0	0	2
EIFs:	1	0	0	1
EIs :	2	0	0	2
EOs :	1	1	0	2
EQs :	0	0	1	1

FFP - Function Types:

DETs:	1-19	20-50	>51	#
ICRs:	0	0	0	0
ICWs:	0	0	0	0
ECXs:	0	0	0	0
ECEs:	0	0	0	0

Multiple occurrence groups of data:

	l	a	h	#
UCG:	0	0	0	0
RCG:	0	0	0	0

Single occurrence groups of data:

UCG : 0 DETs  
 RCG : 0 DETs

-----

Type : Process  
 Name : DetailProjForm  
 Application : FPC  
 Section : A.2.1  
 Number of Pages : 0

## Number of functions

-----

## FPA - Function Types:

	l	a	h	#
ILFs:	0	0	0	0
EIFs:	0	0	0	0
EIs :	2	0	0	2
EOs :	1	0	0	1
EQs :	0	0	0	0

## FFP - Function Types:

DETs:	1-19	20-50	>51	#
ICRs:	0	0	0	0
ICWs:	0	0	0	0
ECXs:	0	0	0	0
ECEs:	0	0	0	0

## Multiple occurrence groups of data:

	l	a	h	#
UCG:	0	0	0	0
RCG:	0	0	0	0

## Single occurrence groups of data:

UCG : 0 DETs  
RCG : 0 DETs

-----  
.  
.  
.

**fpc.sta**

\*\*\*\*\*

Function Point Counter  
Project Statistic File

\*\*\*\*\*

Projectname : FPC  
Unadjusted Function Points : 382  
Versions : 1  
Applications : 2

Processess : 23  
 Duration (days) : 272  
 Costs (\$) : 5000  
 Counted Pages : 11

Costs / Function Point (\$) : 13  
 Pages / Processes : 0  
 Function Points / Day : 1  
 Function Points / Process : 16  
 Function Points / Page : 34

Function Points FPA:  
 (Data & Transactional Function Types)

	ILF	EIF	EI	EO	EQ	#
FPs:	87	27	99	115	54	382

Function Points FFP : (Transactional Function Types)

	ICR	ICW	ECX	ECE	#
FPs:	0	0	0	0	0

(Data Function Types)

	s.o.UGC	s.o.RCG	m.o.UGC	m.o.RCG	#
FPs:	0	0	0	0	0

=====

Type : Version  
 Name : Version 1.0

Unadjusted Function Points : 382  
 Applications : 2  
 Processess : 23  
 Duration (days) : 272  
 Costs (\$) : 5000  
 Counted Pages : 11

Costs / Function Point (\$) : 13  
 Pages / Processes : 0  
 Function Points / Day : 1  
 Function Points / Process : 16  
 Function Points / Page : 34

Function Points FPA:  
 (Data & Transactional Function Types)

	ILF	EIF	EI	EO	EQ	#
FPs:	87	27	99	115	54	382

## Function Points FFP : (Transactional Function Types)

	ICR	ICW	ECX	ECE	#
FPs:	0	0	0	0	0

## (Data Function Types)

	s.o.UGC	s.o.RCG	m.o.UGC	m.o.RCG	#
FPs:	0	0	0	0	0

---

Type : Application  
Name : FPC

Unadjusted Function Points : 382  
Processes : 23  
Duration (days) : 182  
Costs (\$) : 3500  
Counted Pages : 11

Costs / Function Point (\$) : 9  
Pages / Processes : 0  
Function Points / Day : 2  
Function Points / Process : 16  
Function Points / Page : 34

## Function Points FPA:

## (Data &amp; Transactional Function Types)

	ILF	EIF	EI	EO	EQ	#
FPs:	87	27	99	115	54	382

## Function Points FFP : (Transactional Function Types)

	ICR	ICW	ECX	ECE	#
FPs:	0	0	0	0	0

## (Data Function Types)

	s.o.UGC	s.o.RCG	m.o.UGC	m.o.RCG	#
FPs:	0	0	0	0	0

---

Type : Process

Name : MainForm

Unadjusted Function Points : 26

Counted Pages : 1

Function Points / Page : 26

Function Points FPA:  
(Data & Transactional Function Types)

	ILF	EIF	EI	EO	EQ	#
FPs:	14	5	3	4	0	26

Function Points FFP : (Transactional Function Types)

	ICR	ICW	ECX	ECE	#
FPs:	0	0	0	0	0

(Data Function Types)

	s.o.UCG	s.o.RCG	m.o.UCG	m.o.RCG	#
FPs:	0	0	0	0	0

Type : Process  
Name : NewProjForm

Unadjusted Function Points : 40

Counted Pages : 1

Function Points / Page : 40

Function Points FPA:  
(Data & Transactional Function Types)

	ILF	EIF	EI	EO	EQ	#
FPs:	14	5	6	9	6	40

Function Points FFP : (Transactional Function Types)

	ICR	ICW	ECX	ECE	#
FPs:	0	0	0	0	0

(Data Function Types)

	s.o.UCG	s.o.RCG	m.o.UCG	m.o.RCG	#

```
-----|-----|-----
FPs: | 0    0    0    0    0 | 0
```

```
-----
Type           : Process
Name           : DetailProjForm
```

```
Unadjusted Function Points : 10
Counted Pages              : 0
```

```
Function Points / Page     : 0
```

```
Function Points FPA:
(Data & Transactional Function Types)
```

```
-----|-----|-----|-----|-----|-----
      | ILF   EIF   EI   EO   EQ   | #
FPs:  | 0     0     6   4   0   | 10
```

```
Function Points FFP : (Transactional Function Types)
```

```
-----|-----|-----|-----|-----
      | ICR   ICW   ECX   ECE   | #
FPs:  | 0     0     0     0   | 0
```

```
(Data Function Types)
```

```
-----|-----|-----|-----|-----
      | s.o.UGC s.o.RCG m.o.UGC m.o.RCG | #
FPs:  | 0     0     0     0   | 0
```

```
-----
.
.
.
```

## B.2 Fazit

Als erstes muss gesagt werden, dass das meine erste Function-Point Zählung nach der FPA-Methode<sup>33</sup> war. Damit sind einige Resultate aus dem Zähl-Prozess sicherlich diskussionswürdig. Allerdings kann man durch den Vergleich der Function-Points der einzelnen Prozesse untereinander, gut auf den Unterschied beim Aufwand zu deren Implementierung schließen (die generierten Dateien, inklusive der hier nicht aufgeführten Tabellen, liegen dem

<sup>33</sup>Als Grundlage für den Zählprozess diente das “Function Point Counting Practice Manual, Release 4.0” ([MRW94]) der “International Function Point User Group (IFPUG)”

Datenträger zur Diplomarbeit bei). Für eine weitere Auswertung der Endresultate fehlen aber Erfahrungen aus ähnlichen Projekten.

Der Hauptgründe für die Zählung der Function-Points mit FPC lagen deshalb auch in der Überprüfung der Applikation unter realistischen Bedingungen und in der Testung der Bedienstrategie, beim Anlegen der Datensätze und der Eingabe der Daten.

## C Ausschnitte aus dem FPC–Sourcecode

### Startroutine von Palm–Applikationen

```

/* Main-Funktion von Palm-Applikationen*/
DWord PilotMain (Word cmd, Ptr cmdPBP, Word launchFlags)
{
    int error;

    if (cmd == sysAppLaunchCmdNormalLaunch)
    {
        error = StartApplication(); /* Application start code */
        if (error) return error;
        EventLoop();                /* Eventbehandlungsroutine */
        StopApplication ();          /* Application stop code */
    }
    return 0;
}

```

### Applikationsinitialisierung bei Ereignis *Programmstart*

```

/* Applikationsinitialisierung */
static int StartApplication(void)
{
    VoidHand      RecHandle;
    Ptr           RecPointer;
    UInt          index = 0, theCardNum, theAttributes;
    struct ConfRec rec;
    Word          err;
    LocalID       theLocalID;

    /*Oeffnen der FPC-Datenbank*/
    FPCntDB = DmOpenDatabaseByTypeCreator(FPCntDBType, FPCntAppID, dmModeReadWrite);
    if (!FPCntDB)
    {
        /* Datenbank nicht vorhanden: Anlegen */
        err = DmCreateDatabase(0, FPCntDBName, FPCntAppID, FPCntDBType, false);
        /* Oeffnen der angelegten Datenbank und kreieren des KonfigRecords*/
        FPCntDB = DmOpenDatabaseByTypeCreator(FPCntDBType, FPCntAppID, dmModeReadWrite);
        RecHandle = DmNewRecord(FPCntDB, &index, sizeof(struct ConfRec));
        RecPointer = MemHandleLock(RecHandle);
        rec.FirstProj = rec.NumProj = rec.LastProj = 0;
        rec.FirstStat = rec.NumStat = rec.FirstApp = rec.NumApp = 0;
        rec.FirstProc = rec.NumProc = rec.LastID = 0;
        /* Schreiben des Records in die Datenbank*/
        err = DmWrite(RecPointer, 0, &rec, sizeof(struct ConfRec));
        MemPtrUnlock(RecPointer);
        DmReleaseRecord(FPCntDB, index, true);
        /* Setzen von Datenbankattributen */
        DmOpenDatabaseInfo(FPCntDB, &theLocalID, NULL, NULL, &theCardNum, NULL);
        DmDatabaseInfo(theCardNum, theLocalID, NULL, &theAttributes, NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL);
        theAttributes |= dmHdrAttrBackup;
        DmSetDatabaseInfo(theCardNum, theLocalID, NULL, &theAttributes,

```

```

        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);
    }
    /* History enth{"a}lt die Anzahl und die Nummern der ge{"o}ffneten Forms */
    history[0] = 0;
    history[1] = 0;
    /* Schliessen der Datenbank */
    DmCloseDatabase(FPCntDB);
    /* Gehe zum Startbildschirm der Applikation */
    FrmGotoForm(FctPtsMainForm);
}

```

## Routine zur Annahme von Events und zur deren Verteilung an den entsprechenden Eventhandler

```

/* Eventbehandlungsroutine */
static void EventLoop(void)
{
    Short    err;
    Int      formID;
    FormPtr  form;
    EventType event;

    do
    {
        /* Annahme eines Events und Speicherung in einer Datenstruktur */
        EvtGetEvent(&event, 200);
        if (SysHandleEvent(&event)) continue;
        if (MenuHandleEvent((void *)0, &event, &err)) continue;
        /* Verteilung des eingegangenen Events an den enstprechenden Handler, */
        /* abhaenig in welcher Form das Ereignis passiert ist */
        if (event.eType == frmLoadEvent)
        {
            formID = event.data.frmLoad.formID;
            form = FrmInitForm(formID);
            FrmSetActiveForm(form);
            switch (formID)
            {
                case FctPtsMainForm:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) MainFormHandler);
                    break;
                case NewForm:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) NewFormHandler);
                    break;
                case DetailForm:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) DetailFormHandler);
                    break;
                case NewProcForm:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) NewProcFormHandler);
                    break;
                case EnterDataFPA1Form:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) EnterDataFPA1FormHandler);
                    break;
                case EnterDataFPA2Form:
                    FrmSetEventHandler(form, (FormEventHandlerPtr) EnterDataFPA2FormHandler);
                    break;
                case QEnterDataFPAForm:

```

```

        FrmSetEventHandler(form, (FormEventHandlerPtr) QEnterDataFPFormHandler);
        break;
    case EnterDataFFP1Form:
        FrmSetEventHandler(form, (FormEventHandlerPtr) EnterDataFFP1FormHandler);
        break;
    case EnterDataFFP2Form:
        FrmSetEventHandler(form, (FormEventHandlerPtr) EnterDataFFP2FormHandler);
        break;
    case QEnterDataFFPForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) QEnterDataFFPFormHandler);
        break;
    case ValInputForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) ValInputFormHandler);
        break;
    case EntryListForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) EntryListFormHandler);
        break;
    case FPsForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) FPsFormHandler);
        break;
    case FPsFPForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) FPsFPFormHandler);
        break;
    case FPsFFPForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) FPsFFPFormHandler);
        break;
    case AnalyzeForm:
        FrmSetEventHandler(form, (FormEventHandlerPtr) AnalyzeFormHandler);
        break;
    }
}
FrmDispatchEvent(&event);
} while(event.eType != appStopEvent);
}

```

## Eventhandler des FPC-Startbildschirms

```

/* Eventhandler fuer die FctPtsMainForm */
static Boolean MainFormHandler(EventPtr event)
{
    FormPtr      form;
    Int          handled = 0;
    struct ProjStat *proj;

    switch (event->eType)          /* Auswahl der Routine nach eingegangenem Event */
    {
    case frmOpenEvent:             /* Event: Oeffnen der FctPtsMainForm */
        form = FrmGetActiveForm(); /* Hole die Nummer des aktiven Bildschirms */
        FrmDrawForm(form);        /* Zeichnen der Form */
        handled = 1;
        break;
    case ctlSelectEvent:          /* Ein Button wurde gedrueckt */
        NewFormFlag = 0;
        history[0] = 1;
        atwork = 0;
        if (event->data.ctlSelect.controlID == 1101) /* Create New Project - Button */
        {

```

```

        history[1] = NewForm;
        FrmGotoForm(history[history[0]]);
    }
    if (event->data.ctrlSelect.controlID == 1102) /* Modify Project - Button */
    {
        history[1] = EntryListForm;
        EntryFlag = 0;
        FrmGotoForm(history[history[0]]);
    }
    if (event->data.ctrlSelect.controlID == 1103) /* Analyze Project - Button */
    {
        history[1] = EntryListForm;
        EntryFlag = 1;
        FrmGotoForm(history[history[0]]);
    }
    handled = 1;
    break;
}
return handled;
}

```

Funktion zum Auffinden von Records in der Datenbank anhand der Datensatz-ID, unter Verwendung des Algorithmus Binaeres Suchen

```

/* Funktion zum Finden von Recordindizes anhand der ID */
UInt GetRecordIndex(Int typ, Int id)
{
    struct DBRec      *gi_dbrec;
    struct TreePtr    *gi_treeptr;
    struct ProjStatRec *gi_projstat;
    struct AppRec     *gi_app;
    struct ProcRec    *gi_proc;
    struct ConfRec    *gi_conf;
    UInt              l, r, x, v;
    VoidHand          gi_RecHandle;

    /* Oeffnen der Datenbank */
    OpenDatabase();
    gi_RecHandle = DmGetRecord(fpcdb, ConfIdx);
    gi_conf = MemHandleLock(gi_RecHandle);
    /* Initialisierung von l und r mit dem Index des ersten und letzten Records */
    /* Abh{"a"}nig vom Recordtyp */
    switch (typ)
    {
    case ProjTyp:
        l = gi_conf->FirstProj;
        r = l + gi_conf->NumProj - 1;
        break;
    case StatTyp:
        l = gi_conf->FirstStat;
        r = l + gi_conf->NumStat - 1;
        break;
    case AppTyp:
        l = gi_conf->FirstApp;
        r = l + gi_conf->NumApp - 1;

```

```

    break;
case ProcTyp:
    l = gi_conf->FirstProc;
    r = l + gi_conf->NumProc - 1;
    break;
}
MemHandleUnlock(gi_RecHandle);
DmReleaseRecord(fpcdb, ConfIdx, false);
/* Binaeres Suchen zum Auffinden von Prozessen */
if (typ == 3)
{
    while (r >= l)
    {
        x = (l + r) / 2;
        gi_RecHandle = DmGetRecord(fpcdb, x);
        gi_proc = MemHandleLock(gi_RecHandle);
        if (id == gi_proc->id)
        {
            MemHandleUnlock(gi_RecHandle);
            DmReleaseRecord(fpcdb, x, false);
            DmCloseDatabase(fpcdb);
            return (x);
        }
        if (id < gi_proc->id) r = x - 1;
        else l = x + 1;
        MemHandleUnlock(gi_RecHandle);
        DmReleaseRecord(fpcdb, x, false);
    }
}
/* Binaeres Suchen zum Auffinden von Projekten, Versionen, Applikationen */
/* Die Recordtypen haben aehnlichen Aufbau */
else
{
    while (r >= l)
    {
        x = (l + r) / 2;
        gi_RecHandle = DmGetRecord(fpcdb, x);
        if ((typ == 0) || (typ == 1) )
        {
            gi_projstat = MemHandleLock(gi_RecHandle);
            gi_dbrec = &gi_projstat->dbrec;
        }
        else
        {
            gi_app = MemHandleLock(gi_RecHandle);
            gi_dbrec = &gi_app->dbrec;
        }
        if (id == gi_dbrec->id)
        {
            MemHandleUnlock(gi_RecHandle);
            DmReleaseRecord(fpcdb, x, false);
            DmCloseDatabase(fpcdb);
            return (x);
        }
        if (id < gi_dbrec->id) r = x - 1;
        else l = x + 1;
        MemHandleUnlock(gi_RecHandle);
        DmReleaseRecord(fpcdb, x, false);
    }
}

```

```
    }  
    DmCloseDatabase(fpcdb);  
}
```

## Literatur

- [Abr99] A Abran, Herausgeber. *FFP Release 2.0: An Implementation of COSMIC Functional Size Measurment Concepts*, 1999. Proceedings of FESMA '99, Amsterdam, Oct. (1999).
- [ADS<sup>+</sup>99] A. Abran, J.M. Desharnais, Oligny S., D. St-Pierre und C.R. Symons, Herausgeber. *COSMIC-FFP Measurment Manual version 2.0*, 1999. Ed. S. Oligny, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, Downloadable at <http://www.lrgl.uqam.ca/ffp.html>, September (1999).
- [Bal98] Helmut Balzert. *Lehrbuch der Softwaretechnik*. Spektrum Akademischer Verlag, Heidelberg – Berlin, 1998.
- [BF00] K. Bundschuh und A. Feby. *Aufwandschätzung von IT-Projekten*. MITP-Verlag, Bonn, 2000.
- [Car98] Jeff Carlson. *Palm III and PalmPilot*. Peachpit Press, 1998.
- [DA99] Reiner Dumke und Alain Abran. *Software Measurment Current Trends in Research and Practice*. DUV Deutscher Universitäts Verlag, 1999.
- [DFKW96] Reiner Dumke, Erik Foltin, Reinhard Koeppel und Achim Winkler. *Softwarequalität durch Meßtools*. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1996.
- [DFS99] Reiner Dumke, Erik Foltin und Ulrich Schweikl, Herausgeber. *Applicability of Full Function Points for Siemens AT*, 1999. Proceedings of IWSM '99, Proceedings are downloadable at <http://www.lrgl.uqam.ca/iwsm99/index2.html>, September (1999).
- [Dum92] Reiner Dumke. *Softwareentwicklung nach Maß Schätzen - Messen - Bewerten*. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1992.

- [Dum00] Reiner Dumke. *Software Engineering Eine Einführung für Ingenieure: Systeme, Erfahrungen, Methoden, Tools*. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2. Auflage, 2000.
- [EP98] Hans-Erik Eriksson und Magnus Penker. *UML Toolkit*. John Wiley & Sons, Inc., 1998.
- [Lew91] T.G. Lewis. *CASE: Computer Aided Software Engineering*. Van Nostrand Reinhold, New York, 1991.
- [Mah92] Hans Mahnke. *Projektmanagement unter Windows mit MS-PROJECT 3.0*. Vogel Verlag und Druck Kg, Würzburg, 1992.
- [Mar94a] John J. Marciniak, Herausgeber. *Encyclopedia of Software Engineering, Volume 1 A-N*. John Wiley & Sons, Inc, 1994.
- [Mar94b] John J. Marciniak, Herausgeber. *Encyclopedia of Software Engineering, Volume 2 O-Z*. John Wiley & Sons, Inc, 1994.
- [MRW94] Frank Mazzucco, Robin Ragland und Gary Walker, Herausgeber. *Function Point counting Practices Manual, Release 4.0*, 1994. The International Function Point User Group (IFPUG).
- [OG97] Östen Oskarsson und Robert Glass. *ISO 9000 und Softwarequalität*. Prentice-Hall Verlag GmbH, 1997.
- [Pfl99] Shari Lawrence Pfleeger. *Software Engineering Theory and practice*. Prentice-Hall, Inc., 1999.
- [PJ91] Meilir Page-Jones. *Praktisches DV-Projekt-Management*. Carl Hanser Verlag, München – Wien, 1991.
- [Pog98] David Pogue. *PalmPilot The Ultimate Guide*. O'Reilly & Associates, Inc., 1998.
- [RMS99] Neil Rhodes, Julie McKeehan und Mark Stone. *Palm OS Programming : The Developer's Guide*. O'Reilly & Associates, Inc, 1999.
- [Sed94] Robert Sedgewick. *Algorithmen*. Addison-Wesley (Deutschland) GmbH, 1994.

- [She81] B. A. Sheil. *The Psychological Study of Programming*. Computer Surveys, 1981.
- [SPMA<sup>+</sup>97] D. St-Pierre, M. Maya, A. Abran, J.M. Desharnais und P. Bourque, Herausgeber. *Full Function Points: Function Points Extensions for Real-Time Software - Counting Practice Manual*, 1997. Technical Report no. 1997-04, Software Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, Downloadable at <http://www.lrgl.uqam.ca/ffp.html>, September (1997).
- [SvG00] Jochen Seemann und Jürgen Wolff von Gudenberg. *Software-Entwurf mit UML*. Springer Verlag, Berlin – Heidelberg, 2000.
- [Sym99] C.R. Symons, Herausgeber. *COSMIC aims, desing principles and progress*, 1999. Proceedings of IWSM '99, Proceedings are downloadable at <http://www.lrgl.uqam.ca/iwsm99/index2.html>, September (1999).
- [Tha97] Richard H. Thayer, Herausgeber. *Software Engineering Project Management*. IEEE Computer Society Press, 1997.

## **Selbständigkeitserklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 30. Januar 2001

Daniel Reitz

## Thesen

1. Mit der vorliegenden Arbeit wurde ein CASE-Tool zur Unterstützung des Function-Point Zählprozesses realisiert. Das Programm entspricht weitestgehend den gestellten Anforderungen.
2. Es ist generell möglich den Palm-Handheldcomputer zur Unterstützung des Softwareentwicklungsprozesses einzusetzen.
3. Es ist nur sinnvoll CASE-Tools für den Palm zu entwickeln, wenn die Vorteile des Palms, bei der Unterstützung der entsprechenden Methode, die Nachteile überwiegen.
4. Im aktuell existierenden Softwareangebot für den Palm, wird der CASE-Bereich kaum berücksichtigt.
5. Die Einschränkungen bei der Eingabe größerer Datenmengen, können durch die Anschaffung einer mobilen Palm-Tastatur umgangen werden.
6. Die in dieser Arbeit vorgestellten CASE-Tools für den Palm sind alle für den stand-alone Betrieb konzipiert worden. Eine Umstellung dieser Strategie auf den Einsatz der Werkzeuge, als Frontend für Desktop-basierte CASE-Tools, würde ein Großteil der technologischen Einschränkungen der Palm-Plattform abschwächen. Die Desktop-Applikation müsste somit um Möglichkeiten zur Eingabe der Daten, zur Verwaltung der Datenbank und zur Präsentation der Resultate erweitert werden. Das Palm-Programm würde in seiner Ausstattung nicht eingeschränkt, bräucht aber nur die gerade benötigten Daten enthalten und müsste den Datenbestand mit der Desktop-Applikation synchronisieren können.
7. Die in dieser Arbeit getroffenen Aussagen über Einschränkungen des Palms, werden mit kommenden Generationen von Handheld-Computern teilweise oder komplett revidiert werden müssen. Gerade was der Speicherplatz und die Rechenleistung angeht, besitzt die aktuelle Palm-Konkurrenz schon ein wesentlich höheres Potential.
8. Die Einführung von Spracherkennungstechniken im Bereich der Handheldcomputer, wird die Eingabe von größeren Datenmengen ermöglichen.

9. Der Einsatz des Internets zur Synchronisation von Palm-Datenbanken mit dem PC, erfordert zwingend die Verwendung von Verschlüsselungs-Mechanismen zur Übertragung sensibler Daten.