Diplomarbeit

Realisierung einer sicheren Mobilen Agenten Plattform auf einem PDA

ROLF GRINDEL

Fachhochschule Bingen
Fachbereich Elektrotechnik
Studiengang Angewandte Informatik
Berlinstraße 109, 55411 Bingen

Fraunhofer Institut für
Graphische Datenverarbeitung
Abteilung Sicherheitstechnologie
für Graphik- und Kommunikationssysteme
Rundeturmstraße 6, 64283 Darmstadt

Diplomarbeit

Prüfer	Prof. Dr.-Ing. Peter Rausch Fachhochschule Bingen
Betreuer	Dipl.-Ing. (FH) Ulrich Pinsdorf Fraunhofer IGD, Darmstadt
Kandidat	Rolf Grindel Matr.-Nr. 187631
Tag der Ausgabe	19. Juni 2001
Tag der Abgabe	15. Februar 2002

Inhaltsverzeichnis

I Grundlagen **1**

1 Einleitung **3**

1.1 Motivation 3

1.2 Zielsetzung 4

1.3 Aufbau der Diplomarbeit 5

2 Mobile Agenten **7**

2.1 Definition 7

2.2 Agentensysteme 8

2.3 Verschiedene Agententypen 10

3 Das SeMoA - Projekt **13**

3.1 Was ist SeMoA ? 13

3.2 Vorraussetzungen für den Betrieb von SeMoA 15

3.2.1 Multithreading 16

3.2.2 Thread Groups 16

3.2.3 Complete Garbage Collection 17

3.2.4 Object Serialization 17

3.2.5 Dynamic Class Loading 18

3.2.6 Zipfile support 18
3.2.7 Just-In-Time Compiler 18
3.2.8 Dynamic Proxy Generation 19
3.2.9 Java Security Framework 19

4 Testumgebung **21**
4.1 Server im Sendemodus 22
4.2 Server im Empfangsmodus 24
4.3 Agent 26
4.4 Zusammenfassung 27

II Palm Vx **29**

5 Palm Vx - Das Gerät **31**
5.1 Technische Details 32
5.2 Erweiterungsmöglichkeiten 32
5.3 Die Zukunft des Palm 34

6 Virtual Machines für Palm **37**
6.1 Sun Java2 Platform Micro Edition 37
6.2 IBM Visual Age Micro Edition 42
6.3 Kada Mobile KadaVM 44
6.4 Übersicht untersuchter Virtual Machines für den Palm 48

7 Mobile Agentenplattformen auf dem Palm **49**
7.1 LEAP 49
7.2 SeMoA 50

III Compaq iPAQ **53**

8 iPAQ 3660 - Das Gerät **55**

8.1 Technische Details 56

8.2 Erweiterungsmöglichkeiten 58

8.3 Die Zukunft des iPAQ 59

9 Virtual Machines für iPAQ **61**

9.1 Java Virtual Machines für WindowsCE 61

9.1.1 Jeode 61

9.1.2 KadaVM 62

9.2 Java Virtual Machines unter LinuxARM 63

9.2.1 Kaffe 64

9.2.2 Blackdown JRE 1.3.1RC1 64

9.3 Java als OS - SavaJe XE 68

9.4 Übersicht untersuchter Virtual Machines für den iPAQ 71

10 Mobile Agentenplattformen auf dem iPAQ **73**

10.1 SeMoA 73

10.1.1 Betrieb 73

10.1.2 Benchmarks 76

10.1.3 Beispiel einer Anwendung 80

10.2 LEAP 87

10.3 Grasshopper 87

IV Resümee **91**

11 Zusammenfassung und Ausblick **93**

11.1 Zusammenfassung 93

11.2 Bewertung der gefundenen Lösung 94

11.3 Ausblick 95

V Anhang **97**

A Palm Vx **99**

A.1 Installation: Sun Microsystems - JavaME 99

A.2 Installation: Kadasystems - KadaVM 100

A.2.1 Virtual Machine 100

A.2.2 Installation der Testumgebung unter KadaVM 103

B Compaq iPAQ **105**

B.1 Installation: Insignia Solutions Jeode 105

B.2 Installation: LISA mLinux 106

B.2.1 Vorraussetzungen 106

B.2.2 Sicherheitskopie von WindowsCE 107

B.2.3 Bootloader 108

B.2.4 Kernel und Filesystem 110

C Handhelds.org Familiar **113**

C.1 Installation 113

C.1.1 Vorraussetzungen 113

C.1.2 Sicherheitskopie von WindowsCE 114

C.1.3 Bootloader 115

C.1.4 Kernel und Root-Filesystem 117

C.2 Aironet 4500 Wireless LAN-Karte 119

C.3 Blackdowns Java Runtime 1.3.1RC1 für Linux 120

C.4 Installation der Testumgebung 121

C.5 SeMoA 121

C.6 Erzeugen eigener JFFS2-Flashabbilder 122

C.7 Inhalt des ipaq-familiar_0.5-semoa.jffs2-Flashimages 123

D SavaJe XE **129**

D.1 Bootloader 129

D.2 Partionieren und Laden der Partitionen 130

E Restauration WindowsCE **133**

E.1 Vorraussetzungen 133

E.2 Restauration von WindowsCE (neue Methode) 134

E.2.1 Durchführung 134

E.2.2 Überprüfung der Restauration 134

E.2.3 Wiederherstellen des original Bootloaders 135

E.3 Restauration von WindowsCE (alte Methode) 136

Abbildungsverzeichnis

3.1 Sicherheitsarchitektur von SeMoA 14

4.1 GUI-Komponente des Agenten 27

5.1 Palm Vx 31

6.1 Java Virtual Machines von Sun Microsystems 38

6.2 KVM (Java2ME) auf dem Palm Vx 39

6.3 Beziehung von J2SE, CDC und CDLC 41

6.4 IBM Visual Age Micro Edition (VAME) auf dem Palm Vx 43

6.5 KadaVM auf Palm Vx 45

6.6 POSE-Fehlermeldung beim Serialisieren von Objekten (KadaVM) 46

8.1 Compaq iPAQ 3660 55

9.1 LISA mLinux 0.9 auf dem iPAQ 65

9.2 SeMoA auf iPAQ mit Familiar 67

9.3 SavaJe XE - Anwendungsbereich 68

10.1 Passwortaufforderung von SeMoA auf dem iPAQ 74

10.2 JumpingAgent von SeMoA 75

10.3 Klassendiagramm: MP3-Agent 83

10.4 MP3-Player GUI: Browse Option 85

10.5 MP3-Player GUI: Fetch Option 86

10.6 LEAP auf dem iPAQ 88

A.1 Conduit Configuration 102

A.2 Konfiguration des KadaInstaller-Conduit 102

A.3 KadaInstaller 104

Tabellenverzeichnis

5.1 Technische Details — Palm Vx 33

5.2 Stärken und Schwächen — Palm Vx 34

6.1 Eigenschaften der Virtual Machines für Palm 48

8.1 Technische Details — Compaq iPAQ 3660 57

8.2 Stärken und Schwächen — iPAQ 58

9.1 Stärken und Schwächen — LinuxARM 63

9.2 Eigenschaften der Virtual Machines für iPAQ 71

10.1 Technische Daten der am Benchmark beteiligten Rechner 77

10.2 Benchmark-Ergebnis: „Arrakis“ und „Giedi“ 78

10.3 Benchmark-Ergebnis: „Arrakis“ und „Laptop“ 78

10.4 Benchmark-Ergebnis: „Laptop“ und iPAQ 79

C.1 Tabelle C.1: Paketliste für Familiar 0.5 128

Danksagung

Ich möchte an erster Stelle meinen Eltern Annemarie und Helmut Grindel danken, daß sie mich in meinem Tuen unterstützt und mir das Studium der Angewandten Informatik ermöglicht haben. Auch möchte ich meinem Bruder Horst Grindel danken, für die Bereitstellung von Hardware und Ressourcen mit denen der Datenaustausch wesentlich erleichtert wurde.

Desweiteren möchte ich mich bei Dipl.-Ing. (FH) Ulrich Pinsdorf für die hervorragende Betreuung bedanken und dafür, daß er mir dieses Thema angeboten hat und stets mit Ratschlag und konstruktiver Kritik zu helfen wußte.

Auch gilt mein Dank allen weiteren Mitarbeitern des Fraunhofer Instituts, die mich bei meiner Arbeit unterstützen und ein angenehmes Arbeitsklima geboten haben.

Für die Betreuung der Diplomarbeit seitens der Fachhochschule Bingen, sowie für wertvollen Tips und konstruktive Kritik, möchte ich Prof. Dr.-Ing. Peter Rausch danken.

Ferner danke ich meinen ehemaligen Studienkollegen Werner Beutel, Markus Rudolf, sowie Thomas Hottum für anregende Diskussionen, nützliche computertechnische Hinweise, insbesondere im Bezug auf Vorgehensweise und LaTeX.

Zuletzt danke ich all denen, die mich während der Entstehungszeit dieser Arbeit unterstützt und für die nötige Ablenkung gesorgt haben, besonders meinen Freunden Isabell von Harder-Roth, Jens von Harder und Dipl.-Ing. Walter Horn.

Teil I

Grundlagen

Kapitel 1

Einleitung

1.1 Motivation

PDAs[1] sind aus einem modernen Büro oder Arbeitsleben nicht mehr wegzudenken. Waren sie in ihren Anfängen noch stark in Handhabung und Funktionalität eingeschränkt, so profitieren auch sie von der stetigen, leistungssteigernden und miniaturisierenden Entwicklung im Mikroprozessor- sowie im Elektronikbereich. Zu ihren eigentlichen Aufgaben – die Termin-, Adreß- und Aufgabenverwaltung – die sie von ihren nicht-elektronischen Kollegen, den Organizern, geerbt haben, kamen immer mehr Aspekte hinzu. So entwickelten sich die PDAs allmählich zu einem mobilem Büro. Sie erfüllen bzw. übertreffen schon heute die Leistungsklasse früherer Portable PCs oder Laptops. Hauptsächlich verdanken sie diese Wandlung den Fortschritten in der modernen Speichertechnik — hatten die ersten PDAs wenige Kilobyte freien Speicher zur Verfügung, so bieten heutige Geräte beispielsweise 64 MByte RAM. Auch nichtflüchtiger Speicher steht ihnen zur Verfügung — beispielsweise 1 GByte Festplatte via Microdrive — und alle bieten entweder eigenständig oder in Zusammenarbeit mit einem Handy Zugang zum Internet. Was liegt denn nun näher, als moderne Denkansätze und neue Paradigmen, die bisher ihren stationären Kollegen vorbehalten waren, auf diese Klasse von Geräten zu portieren?

[1] Personal Digital Assistant

Mobile Agenten sind ein solches Paradigma und erfreuen sich zunehmender Beliebtheit. Ähnlich ihren menschlichen Kollegen, führen sie, in Vertretung des Anwenders, bestimmte Aufgaben an einem oder mehreren Orten aus. Sie können bei Terminplanungen helfen, als Nachrichtendienste fungieren, gezielt Informationen suchen und zusammenstellen, Daten transportieren, komplexe Berechnungen auf leistungsstarke Systeme auslagern und viels mehr. Um diese Agentenprogramme in kontrollierter Umgebung auszuführen, bedarf es einer Plattform, die den Agenten bestimmte Dienstleistungen, wie den Zugriff auf Ressourcen, Kommunikation mit anderen Agenten, u.a., bereitstellt. Interessant wird es, wenn diese Plattform selbst eine gewisse Mobilität erhält, d.h. daß ihr Benutzer sie, wie auch seinen PDA, „in die Tasche“ stecken kann. Was bringt ihm auf Geschäftsreise ein Agent, der seine Ergebnisse ins Büro schickt und damit die Informationen unter Umständen schwer erreichbar für den Benutzer sind? Der Agent könnte die Ergebnisse und damit auch sich selbst direkt auf den PDA des Benutzers schicken, wenn dort eine solche portable Mobile Agentenplattform laufen würde.

1.2 Zielsetzung

Ziel der Diplomarbeit ist es, eine an die SeMoA[2]-Architektur angelehnte Plattform, oder SeMoA selbst, für mobile Endgeräte (am Beispiel eines PDA) bereitzustellen, auf der Mobile Agenten ausgeführt werden können. Eine konkrete Anwendung ist die Erweiterung der bisherigen Anwendungsbereiche von mobilen Agenten auf mobile Endgeräte (PDA, Handys,...). Als PDA kommen hier ein Palm Vx und Compaq iPAQ 3660 zum Einsatz. Um eine Plattform für Mobile Agenten auf den Handheld PCs zu realisieren, muß eine geeignete Virtual Machine gefunden werden, die allen nötigen Anforderungen in soweit genügt, daß eine Realisierung der Plattform mit geringem Aufwand, etwa durch Bereitstellen fehlender Klassen und Methoden oder durch entsprechend andere Realisierung fehlender Features möglich ist. Um ungeeignete Virtual Machines ausschließen zu können soll eine Testumgebung entwickelt werden. Diese Testumgebung muß essentielle Funktionen, die der Betrieb von Agentensystemen vorraussetzt, auf Funktionalität prüfen. Es sollen Standard Virtual Machines verwendet werden, die von verschiedenen

[2]Secure Mobile Agents

Herstellern angeboten werden. D.h. es soll nach Möglichkeit keinerlei Veränderung an den Virtual Machines selbst erfolgen, anders als bisher realisierte Plattformen, die spezielle Virtual Machines benötigen. Die realisierte Plattform soll den größtmöglichen Funktionsumfang der verwendeten Basisplattform bereitstellen. Um die Möglichkeiten der umgesetzten Plattform zu demonstrieren, soll zudem noch eine präsentationstaugliche Anwendung geschrieben werden, die ein mögliches Einsatzgebiet der gefundenen Lösung demonstriert.

Die Lösung des gestellten Problemkomplexes, Entwicklung des Agentensystems, dessen nötige Erweiterung für mobile Endgeräte, sowie die Entwicklung möglicher Anwendungsbeispiele erfolgt mit Methoden, Konzepten und Werkzeugen des *Objektorientierten Software Engineerings*. Es wird großen Wert auf einen sauberen Entwurf gelegt, da Erweiterungen des Systems, sowie eventuelle Entwicklungen für andere mobile Endgeräte möglich sein sollen. Die Entwicklung eines Agentensystems für einen PDA ist als Beispiel zu sehen. Aus den gewonnen Erkenntnissen soll es möglich sein, auch für andere mobile Endgeräte ein entsprechendes System entwickeln zu können.

1.3 Aufbau der Diplomarbeit

Dieses Dokument gliedert sich in fünf Teile: *Grundlagen*, *Palm Vx*, *Compaq iPAQ*, *Resümee* und *Anhang*.

Im ersten Teil *Grundlagen* wird die Basis geschaffen, auf die der Rest der Diplomarbeit aufbaut. Hier findet sich eine Definition des Begriffs Mobiler Agent, sowie ein Anwendungsbeispiel. Desweiteren wird dort das SeMoA-Projekt vorgestellt, welches im Rahmen der Diplomarbeit seine Anwendung findet.

Der zweite Teil *Palm Vx* stellt das Gerät selbst sowie die Möglichkeiten einer Umsetzung des oben erwähnten SeMoA-Projekts für den Palm vor. Zunächst wird auf die technischen Details des Palm Vx eingegangen, sowie seine Möglichkeiten zur Erweiterung. Anschließend werden die zur Verfügung stehenden Java Virtual Machines in Bezug auf Tauglichkeit untersucht. Am Ende dieses Teils werden kurz weitere Projekte vorgestellt, die sich mit mobilen Agentenplattformen beschäftigen.

Der dritte Teil *Compaq iPAQ* spiegelt die Bemühungen wieder, das SeMoA-Projekt auf eine neuere Generation von Handheld-PCs zu portieren. Hier werden ebenfalls mehrere Java Virtual Machines und ihre Stärken und Schwächen miteinander verglichen. Zusätzlich wird hier noch der Einsatz verschiedener Betriebssysteme beschrieben.

Eine Zusammenfassung der zuvor beschriebenen Ergebnisse findet sich im Teil *Resümee*.

Im *Anhang* finden sich Installationsanleitungen zu den in Teil 2 und 3 verwendeten Programmen und Projekten.

Es sei darauf hingewiesen, daß sich unter den im Rahmen dieser Arbeit verwendeten Bezeichnungen und Begriffen geschützte Markennamen befinden können. Diese werden jedoch, der besseren Lesbarkeit halber, nicht extra gekennzeichnet.

Kapitel 2

Mobile Agenten

In diesem Kapitel wird der Begriff des *Mobilen Agenten* näher definiert, sowie auf dessen Besonderheiten eingegangen. Es werden dabei Bereiche für Anwendungen aufgezeigt und einige mit Hilfe von Beispielen am Ende des Kapitels erläutert.

2.1 Definition

Der Volksbrockhaus [5] definiert *Agent*[1] wie folgt:

> „**Ag'ent** [lat.] *der*, Vermittler, **1)** der *politische A.* ist für Staaten oder polit. Gruppen tätig (im Nachrichtendienst für geheime, oft illegale Aufgaben usw.) **2)** *Handlungsagent*, selbständiger Kaufmann, der ständig für das Handelsgewerbe eines anderen Geschäfte vermittelt oder abschließt."

Ist der erste Teil der Definition hauptsächlich aus Film und Fernsehen bekannt, so ist diese Definition im Kontext der Informatik wenig zutreffend. Der zweite Teil der Definition paßt besser auf das, was in der Informatik unter einem Agenten verstanden wird. ETZIONI und WELD [7] definieren einen Agenten als eine Einheit, die im Auftrag eines Anderen eine oder mehrere Aufgaben erledigt oder dessen Interessen wahrnimmt — nicht zu verwechseln mit einem Anwalt; dieser vertritt

[1] von lat. *agere*: tun, handeln, wirken

hauptsächlich rechtliche Interessen. Nach WOOLRIDGE und JENNINGS kann ein Software Agent wie folgt definiert werden[2]:

> Ein Software Agent ist ein System, in einer Umgebung gelegen, fähig flexible und autonome Handlungen durchzuführen, um bestimmte Ziele zu erreichen. Agenten besitzen die Eigenschaften *autonom*, *reaktiv*, *proaktiv* und *sozial*
>
> **autonom:** Das System agiert ohne direkten Einfluß von anderen. Es besitzt die Kontrolle über seine Aktionen und internen Zustände
>
> **reaktiv:** Das System nimmt die Umgebung wahr und reagiert auf Veränderungen.
>
> **proaktiv:** Agenten sollten nicht nur auf Gegebenheiten der Umgebung antworten, sondern durch Eigeninitiative zielorientierte Handlungen durchführen.
>
> **sozial:** Agenten sollten miteinander interagieren können.

In der Praxis ist ein Agent eine Software, die eine gestellte Aufgabe autonom erledigt. Somit existieren für verschiedene Aufgaben auch verschiedene Agenten, die sowohl unabhängig von einander, als auch nur gemeinsam ein Problem lösen können.

2.2 Agentensysteme

Ein Agentensystem (oder auch Agentenplattform) ist eine Umgebung mit klar definierten Schnittstellen zur Außenwelt. In dieser Umgebung werden die Agenten ausgeführt, können mit Hilfe von Schnittstellen auf Dienste zugreifen, so ihre Umgebung wahrnehmen, mit ihr kommunizieren oder auf Veränderungen reagieren.

Eine Plattform für Mobile Agenten stellt zudem Möglichkeiten bereit, die es einem Agenten erlauben, auf eigenen Wunsch die Lokalität zu wechseln, d.h. auf ein anderes Wirtssystem zu migrieren. Signalisiert ein Agent den „Wunsch“ zu r

[2] Definition nach WOOLRIDGE und JENNING, entnommen aus [48, 52]

Migration und übergibt er den Kontrollfluß an das Agentensystem, so kann es den Agenten zum Transport bereitmachen[3]. Es serialisiert den Agentencode und die im Agenten gespeicherten Daten und transferiert diese, durch etwaige Filter und Protokolle, zu dem Zielsystem. Diese Filter und Protokolle hängen stark vom verwendeten Agentensystem ab. Hier werden oft Logdateieinträge erstellt, die Agenten mit digitale Signaturen unterzeichnet und verschlüsselt. Entsprechend kann die Gegenseite anhand dieser Meta-Daten die Authentizität eines Agenten prüfen. Dort angekommen, wird der Agent deserialisiert und seine Ausführung fortgesetzt; idealerweise an genau der Stelle, an der sie unterbrochen wurde. Dabei unterscheidet man zwei Arten von Migration, *Weak Migration* und *Strong Migration* [48]:

weak: Agentencode und Variablen werden übertragen. Nach der Übertragung werden Variablenwerte wiederhergestellt und der Agent startet neu oder am Beginn einer bestimmten Methode

strong: Wie Weak-Migration jedoch wird der komplette Ausführungszustand mitübertragen und wiederhergestellt. Der Agent startet an der Stelle, an der er die Ausführung unterbrochen hat.

Ein Agentensystem koordiniert somit den Transport der Agenten sowie deren Verwaltung. Eine weitere Aufgabe des Agentensytems ist die Bereitstellung der oben bereits erwähnten Dienste und Schnittstellen beispielsweise zu Datenbanken oder anderen Ressourcen, die den Agenten zur Verfügung stehen sollen. Agentenplattformen bieten eine einfache Möglichkeit, verschiedene Ressourcen kontrolliert einer breiten Masse von Anwendern zur Verfügung zu stellen. So läßt sich eine effektive Ressourcenteilung für beispielsweise verteiltes Rechnen (ähnlich dem Seti@Home-Projekt[4][2]). Andere, ähnlich aufgebaute Projekte beschreibt GLEICH [12]. Auch die Auslagerung von ressourcenhungrigen Aufgaben von ressourcenschwachen Rechnern, wie PDAs, Handhelds und PocketPCs, sind eine ideale Anwendung für Mobilen Agenten. Eine Anbindung an eine Netzwerkinfrastruktur ist

[3] In Java gibt es keine sichere Möglichkeit Threads zu unterbrechen oder zu stoppen. SeMoA baut daher auf die Kooperation der Agenten. Agenten müssen, um zu migrieren, selbsttätig terminieren.

[4] Dieses Projekt basiert aber auf dem typischen Client-/Serverprinzip. Es holt von einem Server ein Datenpaket zur Analyse und bearbeitet dieses dann lokal. Ergebnisse werden dann an den Server zurückgeschickt. Ein Ansatz auf Basis von Mobilen Agenten ist aber denkbar und sinnvoll.

nur zur Migration der Agenten notwendig. Es ist also keine permanente Verbindung, beispielsweise ins Internet, welche für mobile Endgeräte trotz neuer Technologien nachwievor oft teuer und auch schwer zu realisieren ist, erforderlich ist.

Ein Ziel dieser Diplomarbeit ist es unter anderem, für die zuletzt erwähnte Gruppe von Geräten eine sichere Agentenplattform bereitzustellen. Sicher heißt, daß Agenten (und auch Hostsysteme) nicht sabotierend, spionierend und manipulierend auf andere Agenten (und das Hostsystem, ausser den erlaubten Ressourcen) wirken können. Als Basis für diese Plattform setzt diese Arbeit auf das Agentensystem SeMoA, welches näher in Kapitel 3 erläutert wird, auf.

2.3 Verschiedene Agententypen

Hier werden die beiden grundlegenden Agententypen, die in Zusammenhang mit Mobilität existieren, dargestellt. Man unterscheidet hierbei *stationäre* und *mobile* Agenten.

Stationäre Agenten Dieser Begriff wird nur im direkten Vergleich mit Mobilen Agenten benutzt und bezeichnet die Klasse der Nicht-mobilen Agenten (vgl. [52]). Zu dieser Klasse können auch Agenten, die während ihrer Lebenszeit nur auf genau einem Host ausgeführt werden, gezählt werden.

Mobile Agenten NG beschreibt Mobile Agenten mit folgender Definition:

> „A mobile agent is a software entity that represents a user in an environment (network) and can migrate autonomously from node to node, to perform some computation on behalf of the user.“ [48]

Eine ähnliche Definition leitet PINSDORF für Mobile Agenten ab:

> „Ein Mobiler Agent ist ein Agent, der in seinem Lebenszyklus auf verschiedenen Wirtssystemen zur Ausführung kommen kann.“ [52]

Mobile Agenten behalten, im Unterschied zu herkömmlichen Programmen, die einfach von Rechner zu Rechner kopiert und ausgeführt werden, ihren Zustand bei. Das immer wieder kopierte Programm beginnt bei seiner Ausführung seinen Lebenszyklus immer an der ersten Anweisung. Im Gegensatz dazu startet der Agent idealerweise in genau dem Zustand, den er bei seiner Migration innehatte. Jedoch weiß der Agent zu diesem Zeitpunkt noch nicht, ob seine Migration erfolgreich war.

> „*Migration* ist also das Verschieben eines laufenden Programms auf eine andere Ausführungseinheit. Der Prozesszustand des Programms bleibt dabei erhalten. Dafür wird der aktuelle Programmzustand, also die Menge aller Variablenwerte, die das Programm konstituieren, gespeichert. Diese Eigenschaft heißt *Persistenz*." [52]

Im Idealfall bemerkt ein Agent die von ihm angestrebte Migration nicht. Daher sollte er sich orientieren und seine Umgebung wahrnehmen können. Erst wenn sichergestellt ist, daß die Migration erfolgreich war, sollte der Agent seine Aufgabe fortsetzen.

Das herkömmliche Client/Server-Modell erledigt Arbeiten lokal. Dabei werden notwendige Daten vom Server angefordert und zum lokalen Rechner übertragen. Mit diesem Ansatz lassen sich Arbeiten durch mobile Agenten direkt vor Ort, an dem sie ihre Daten erhalten, erledigen. Viele Probleme lassen sich somit effektiv und bandbreitenschonend mit mobilen Agenten bewältigen.

Typische Anwendungsfälle von Mobilen Agenten sind im folgenden aufgeführt:

- Remote Access auf Hard- und Software. Beispiel: Mobile Agenten können zu entfernten Systemen migrieren und dort Analysen anfertigen und entsprechende Maßnahmen einleiten.
- Informationsbeschaffung: Agenten können auf verschiedenen Zielsystemen Datenbanken durchforsten und präsentieren bei ihrer Rückkehr ihr Ergebnis. Die Zielsysteme können vom Benutzer vorgegeben sein oder mit anderen Agenten ausgehandelt werden.

- Personalisierte Dienste: Hier informieren Agenten den Benutzer über Neuerungen oder Änderungen die in seinem Interessenbereich liegen, in dem sie selbsttätig nach passenden Informationen suchen.
- Datenaustausch: Agenten können zwischen verschiedenen Systemen Daten und Dateien austauschen. Dies läßt sich auch zur Fernwartung beispielsweise durch Einspielen von Service Packs und Bugfixes nutzen.
- Zugriff auf Online-Dienste: Agenten suchen Angebote für den Benutzer, oder handeln Verträge aus.
- Verteiltes Rechnen: Hier führen mehrere Agenten auf verschiedenen Systemen Teile einer rechenintensiven Anwendung aus und kehren dann mit ihrem Teilergebnis zu einem Zielsystem zurück.
- Active Mail: Benachrichtigung von anderen Anwendern mit Möglichkeit auf direkte Reaktion [20]. Beispiel: Terminabsprache oder Raumplanung

Kapitel 3

Das SeMoA - Projekt

3.1 Was ist SeMoA ?

SeMoA[1] ist ein Forschungsprojekt am *Fraunhofer Institut für Graphische Datenverarbeitung, Abteilung Sicherheitsysteme für Graphik- und Kommunikationssysteme, Darmstadt*, mit dem Ziel eine sichere Plattform für Mobile Agenten bereitzustellen. Speziell stellt sie Schutzmechanismen vor folgende Szenarien bereit:

- Angriffen zwischen Agenten
- Angriffen von Agenten auf den Agentenserver (Malicious[2] Agent)
- Angriffen des Agentenservers auf die Agenten (Malicios Host)
- Verändern und Ausspähen von Agenten während des Transports

Die Sicherheitsarchitektur von SeMoA ähnelt einem Schalenmodell (vgl. Abbildung 3.1. Agenten müssen durch alle Schutzschichten bevor SeMoA ihre erste Klasse in die Laufzeitumgebung der Virtual Machine lädt.

Die erste, äußere Schicht, *transport layer*, ist für Sicherheit während des Transports der Agenten verantwortlich. Derzeit wird hier eine Implementation des SSL Protokols verwendet. Die zweite Schicht besteht aus einer *Filterpipeline*, die ein

[1] kurz für „Secure Mobile Agents“

[2] engl.: böswillig, arglistig

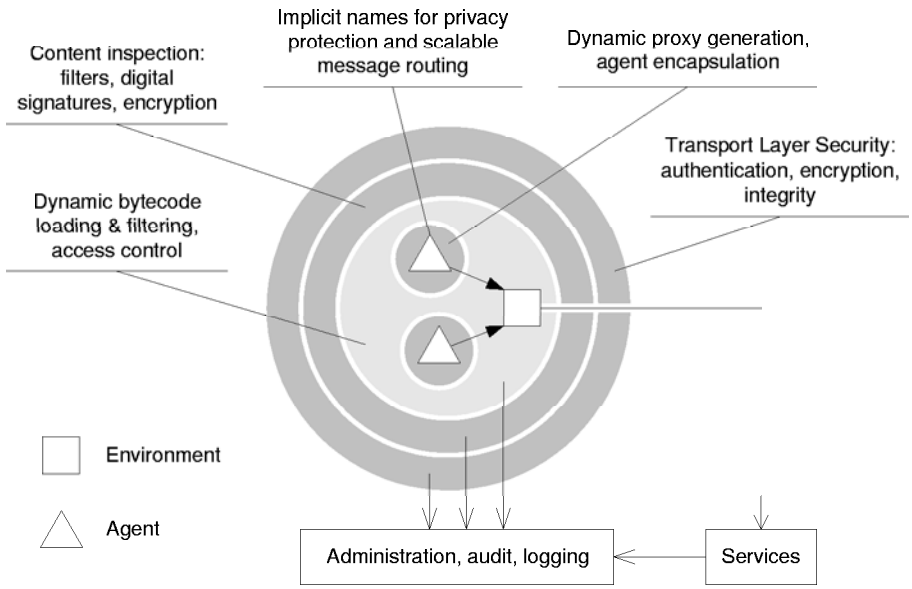

Abbildung 3.1: Sicherheitsarchitektur von SeMoA

Agent beim Ein-/Austreten durchläuft. Jeder Filter inspiziert und bearbeitet jeden Agenten, der durch das System läuft. Der Filter akzeptiert den Agenten oder weist ihn ab. Derzeit enthält SeMoA zwei gegensätzliche Filterpaare, die digitale Signaturen handhaben und selektive Ver-/Entschlüsselung von Agenten durchführen. Zusätzlich ist am Ende der Eingangspipeline ein Filter präsent, der den Agenten Ausführungsrechte zuweist. Als vierte Schicht startet SeMoA jeden Agenten innerhalb einer eigenen Sandbox. Jeder Agent erhält seine eigene Thread Group und einen Classloader. Der Classloader kann alle Klassen laden, die der Agent mit sich bringt. Alle Klassen, die nicht im Klassenpfad stehen, müssen vom Eigentümer des Agenten unterzeichnet sein. Dies stellt die dritte Schicht der Architektur dar.

Auch der Bytecode der Klassen muß beim Laden eine ähnliche Pipeline durchlaufen. So können Filter definiert werden, die den Bytecode prüfen, abweisen oder verändern. SeMoA ist mit einem Filter ausgerüstet, der Klassen bspwe. nach einer `finalize()`-Methode durchsucht. Maliziöse Agenten könnten mit Hilfe dieser Methode den Garbage Collector blockieren und somit die Virtual Machine zum Absturz bringen.

Weitere Informationen werden von ROTH in der Dokumentation, die mit SeMoA ausgeliefert wird, beschrieben [57, 58].

3.2 Vorraussetzungen für den Betrieb von SeMoA

SeMoA selbst ist, wie auch die Agenten die auf der Plattform laufen, vollständig in Java geschrieben. Dies setzt eine Java Virtual Machine auf dem Zielsystem vorraus. Außerdem verwendet SeMoA einige Mechanismen, die erst mit der Einführung von Java2 bereitstehen. Idealerweise sollte eine vollständige Java2 Runtime Edition 1.3.1 Virtual Machine (oder neuer) zur Verfügung stehen.

Notwending für den Betrieb von SeMoA ist, daß die Virtual Machine folgende Eigenschaften beherrscht:

- Multithreading
- Thread Groups
- Complete Garbage Collection
- Exception Handling
- Object Serialization
- Dynamic Class Loading
- Zipfile support
- Just-In-Time Compiler
- Dynamic Proxy Generation
- Java Security Framework

Diese Leistungen der Virtual Machine sind notwendig, um SeMoA unverändert oder mit einigen geringfügigen Änderungen betreiben zu können. Die Bedeutung der einzelnen Punkte und wie sie in SeMoA genutzt werden, wird im folgenden erläutert. Auch wird für den Betrieb einer Mobilen Agentenplattform ein Netzwerkzugang benötigt. Um diesen in Java anzusprechen sollten die `java.net`-Klassen implementiert sein.

3.2.1 Multithreading

Multithreading ist das (quasi-)parallele Ausführen von mehreren Threads auf einem oder mehreren Prozessoren. Threads können synchronisiert oder unabhängig voneinander laufen. So läuft beispielsweise eine GUI[3] in Java immer in einem eigenen Thread, so daß im Hintergrund andere Aufgaben erledigt werden können, wie zum Beispiel das Reagieren auf Eingaben und das Abfangen von Exceptions. Wichtig hierbei ist, daß die Virtual Machine nicht durch eine Anweisung in einem Thread lahmgelegt wird, weil diese Anweisung auf die Reaktion einer Schnittstelle wartet. [34]

Agenten in SeMoA erhalten jeweils ihren eigenen Thread. Somit ist es Aufgabe der Virtual Machine den jeweiligen Agenten Prozessorzeit zuzuordnen und entsprechend zu koordinieren. Desweiteren ist auch die Empfangseinheit ein eigener Thread, sodaß weitere Agenten empfangen werden können, während andere Agenten oder Programme ausgeführt werden.

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

3.2.2 Thread Groups

Die Zusammenfassung von Threads in einer Gruppe wird Thread Group genannt. Thread Groups können weitere Thread Groups enthalten. SeMoA weist jedem Agenten eine eigene Thread Group zu und alle in dem Agenten erzeugten bzw. enthaltenen Threads werden dieser Thread Group hinzugefügt. Mit Hilfe der Thread Group kann erkannt werden, ob die Threads in einer Groupe noch aktiv sind, oder ob sie bereits terminierten. So läßt sich feststellen, ob ein Agent zur Migration bereit ist, da ein kontrolliertes Anhalten von Threads unmöglich ist oder vom Agenten abgefangen werden kann. [34]

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

[3] Graphical User Interface

3.2.3 Complete Garbage Collection

Sobald zu einem Objekt keine Referenzen mehr existieren, sollte es bei dem nächsten Garbage-Collector-Zyklus aus dem Speicher entfernt werden. Der belegte Platz wird dadurch wieder frei und steht für neue Objekte zur Verfügung. Der Garbage Collector muß daher erkennen, ob ein Objekt noch benötigt wird, oder ob der Speicherplatz freigegeben werden kann. Hierzu gibt es verschiedene Ansätze und Algorithmen (z.B. „Boehm-Demers-Weiser"). [41]

Durch die sehr dynamischen Vorgänge bei Mobilen Agenten wäre ohne Garbage Collection der Speicher des Servers sehr schnell voll. Die Virtual Machine und somit auch der Server würde abstürzen oder mit *OutOfMemory* terminieren. Beide müßten periodisch neu gestartet werden. Auch heute noch ist auf PDAs Speicherplatz eine knappe Ressource.

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

3.2.4 Object Serialization

Eine der Kernanforderungen für Mobile Agenten ist die Objektserialisierung und Deserialisierung. Sie ermöglicht es, eine Instanz einer Klasse, sowie darin gespeicherte Variablen und Referenzen, in einen Bitstrom zu wandeln und somit Persistenz zu gewährleisten. Dieser Bitstrom läßt sich dann abspeichern, packen und zu anderen Systemen übertragen. Dort wird der Strom entpackt und deserialisiert, um die übertragene Instanz auf dem Zielsystem zu erzeugen. Sie bildet die Grundlage für den Migrationsvorgang. Ohne Serialisierung läßt sich eine Agentenplattform für Mobile Agenten nicht realisieren. Es ist darauf zu achten, daß Agenten während der Serialisierung keine Veränderungen an ihren Daten vornehmen können, da sonst Dateninkonsistenz auftreten kann. Dies erreicht SeMoA dadurch, daß die Agenten, nachdem sie den „Wunsch" zur Migration geäußert haben, terminieren *müssen* und dann erst serialisiert werden.

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

3.2.5 Dynamic Class Loading

Das dynamische Laden von Klassen (engl.: dynamic class loading) ist eine essentielle Funktion in einem Agentensystem. Da Agenten als kleine Programme realisiert sind und diese fremde Klassen mitbringen, die sie zur Ausführung benötigen. Die mitgebrachten Klassen werden dann über einen speziellen Classloader dynamisch in das Agentensystem / die Virtual Machine geladen. [37]

SeMoA weist jedem Agenten einen eigenen Classloader zu, der zusätzlich zu den im Classpath stehenden Klassen auch agenteneigene Klassen laden kann. Hinzukommt noch, daß die zu ladenden Klassen eine Filterpipeline durchlaufen, die beispielsweise Implementationen von bestimmten Methoden nicht zuläßt, bevor sie in die Virtual Machine geladen werden. [57]

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

3.2.6 Zipfile support

Agenten und ihre Daten können einen großen Umfang annehmen. Zwecks Bandbreitenschonung bietet es sich an, Daten und Agenten mit einem Kompressionsverfahren (verlustfrei oder verlustbehaftet, abhänging von der Datenart) zu packen und erst dann zu verschicken. Dieses Verfahren wurde bereits in den Anfängen der Datenfernübertragung beispielsweise im Fido-Netz[4] verwendet. Dort werden Nachrichten gepackt zwischen den verschiedenen Nodes[5] und Points[6] ausgetauscht.

Dieses Feature ist ein wesentlicher Bestandteil von SeMoA, da die Repräsentation eines Agent aus sicherheitstechnischen Gründen als JAR-Datei erfolgt. Diese Eigenschaft ist daher für SeMoA zwingend erforderlich. [57]

3.2.7 Just-In-Time Compiler

Der Just-In-Time Compiler ist ein optionaler Bestandteil der Virtual Machine von Java. Ohne JIT-Compiler wird der, durch den Java-Compiler erzeugten, Bytecode

[4] Heute noch existent, aber durch das Internet weitgehend abgelöst. [8]

[5] Relais-Stationen und größere Knoten

[6] Endknoten

interpretiert und ausgeführt. Ist ein JIT-Compiler vorhanden, so übergibt die Virtual Machine dem Compiler die `.class`-Datei. Durch den JIT-Compiler wird der Bytecode in plattformabhängigen, ausführbaren Code übersetzt und unmittelbar ausgeführt. Sun Microsystems legt nahe, daß es im allgemeinen schneller sei, den JIT-Compiler zu benutzen, vor allem, wenn bestimmte Methoden immer wieder aufgerufen werden. Diese liegen dann in ausführbarem Code vor und müssen nicht neu interpretiert werden. [44]

Diese Eigenschaft ist für SeMoA nicht zwingend erforderlich, jedoch aus geschwindigkeitstechnischen Gründen empfohlen.

3.2.8 Dynamic Proxy Generation

Proxyobjekte können in vielen Situationen nützlich sein. Sie agieren zwischen einem Client-Objekt und einem Zielobjekt als Relaisstation. Normalerweise liegen Proxyklassen bereits als Java Bytecode vor. Unter Umständen kann es aber nötig sein, den Bytecode von Proxyklassen dynamisch zur Laufzeit zu erstellen.

Eine *dynamische Proxyklasse* ist eine Klasse, die eine, zur Laufzeit spezifizierte, Liste von *Interfaces* implementiert. Jede Instanz einer Proxyklasse ist mit einem Objekt assoziiert, welches die Methoden des Interfaces `InvocationHandler` implementiert. Mit Hilfe von Proxyklassen ist es möglich eine Zugriffskontrolle zu implementieren. Das Konzept *dynamic proxy generation* wurde in der Java Version 1.3 eingeführt.

Diese Eigenschaft ist für SeMoA zwingend erforderlich.

3.2.9 Java Security Framework

Die Java2 Plattform führt, unter anderem, ein neues Sicherheitsmodell ein. Jeder geladenen Klasse werden eine Reihe von Rechten zugewiesen. So kann sehr fein definiert werden, auf welche Ressourcen eine Klasse Zugriff hat und welche Art von Operationen erlaubt sind (z.B. Lesen, Schreiben in bestimmten Verzeichnissen).

> „These new concepts of permission and policy enable the Java2 Platform to offer fine-grain, highly configurable, flexible, and extensible access control. Such access control can now not only be specified for applets, but also for all Java code, including applications, beans, and servlets.“ [65]

SeMoA verwaltet mit Hilfe dieses Mechanismus die Rechte der einzelnen Agenten, ähnlich einer Benutzerverwaltung in einem Netzwerk. So können Agenten von verschiedenen Benutzern auch verschiedene Rechte zugewiesen oder die Ausführung verhindert werden. Daher ist diese Eigenschaft für SeMoA zwingend erforderlich.

Kapitel 4

Testumgebung

Um die grundsätzlichen Fähigkeiten der zur Verfügung stehenden Virtual Machines zu testen, wurde eine Testumgebung entwickelt. Sie prüft essentielle Funktionen, die für ein minimal funktionierendes Agentensystem benötigt werden. Die Umgebung überprüft die nachfolgend aufgezählten Eigenschaften auf Funktionalität.

- Netzwerkunterstüzung auf Basis von TCP/IP
- Dynamisches Laden von Klassen
- Objektserialisierung
- Multithreading
- Unterstützung des ZIP-Formats
- GUI-Komponente (optional)

Diese Komponenten werden in verschiedenen Teilen der Umgebung überprüft. Die beiden zuletzt genannten Punkte, *Unterstützung des Zip-Formats* und *GUI-Komponente*, gelten als nicht kritische Punkte. Sollte eine dieser Funktionen fehlen, so ist ein Agentensystem grundsätzlich realisierbar. Für eine Portierung von SeMoA sind die ersten fünf Komponenten erforderlich.

Im Wesentlichen besteht die Umgebung aus drei Bestandteilen

- einem Server, der einen Agenten lädt und zu einem anderen, im Netzwerk agierenden, Rechner schickt,
- einem Empfänger, der die Verbindungsanfrage des Servers entgegennimmt und den Agenten und seine Daten empfängt und anschließend ausführt,
- einem Agenten, der repräsentativ eine einfache Eingabeaufforderung bildet und die dort eingegebenen Daten transportiert.

Die einzelnen Komponenten sollen nun hinsichtlich ihrer Funktion näher erläutert werden.

4.1 Server im Sendemodus

Nachdem der Server gestartet worden ist, fragt dieser als erstes nach dem Namen der zu ladenden Agentenklasse und anschließend nach einem Hostnamen oder IP-Adresse, zu dem der serialisierte Agent mit seinen Daten geschickt werden soll. Anschließend wird die Agentenklasse dynamisch instanziiert — hier wird der *Dynamic Class Loader* benötigt — und ruft dann die *run()*-Methode des Agenten auf, um diesen auszuführen. Nach der Ausführung dieser Methode und damit auch der Beendigung des Agenten, öffnet der Server eine Socketverbindung zu der angegebenen IP-Adresse. Dann wird der Agent serialisiert, mit einem Standardkompressionsverfahren[1] gepackt und über die Verbindung an den entfernten Rechner geschickt. Wurde alles erfolgreich verschickt, so wird die Socketverbindung wieder geschlossen und der Server beendet.

In dieser Einheit, werden nacheinander vier der genannten Eigenschaften benötigt. Als erstes werden Klassen dynamisch geladen. Da zur Laufzeit erst bekannt wird, wie die Agentenklasse heißt, muß diese dynamisch geladen und ausgeführt werden. Sollte dies nicht möglich sein, so kann ein Agentensystem nicht realisiert

[1] Es wird ZIP verwendet. ZIP ist ein von *PKWARE Inc.* entwickeltes Dateiformat, welches sichere Datenkompression erlaubt. Es wurde bereits 1989 eingeführt und seitdem kontinuierlich erweitert. [53]

werden. Gleiches gilt auch für die zweite getestete Eigenschaft. Der Agent wird zwecks Transport serialisiert. Hierzu wird die Objektserialisierung benötigt. Sollte dafür die Unterstützung in der Virtual Machine fehlen, so kann kein Agentensystem auf dieser Virtual Machine realisiert werden. Nachdem das Objekt serialisiert wurde, wird der Datenstrom mit Hilfe der ZIP-Unterstützung komprimiert. Dies dient hauptsächlich zur Datenreduktion. Sollte diese Einheit fehlen, so müßten Agenten und ihre Daten unkomprimiert verschickt werden. SeMoA benötigt diese Unterstützung, da Agenten aus sicherheitstechnischen Gründen im JAR[2]-Format verschickt werden. JAR ist zum ZIP-Format kompatibel, jedoch erweiterte Sun Microsystems die ZIP-Spezifikation und ermöglichte die Unterstützung für digitale Signaturen innerhalb des Archivs. Als letztes wird der Agent via Netzwerk verschickt. Hierzu wird eine Socketverbindung zu einem angegebenen Rechner hergestellt. Dies benötigt die Unterstützung von TCP/IP durch die Virtual Machine. Um Mobile Agenten zu transportieren muß eine Anbindung an ein bestehendes Netzwerk vorhanden sein. So muß auch die Virtual Machine Mechanismen bereitstellen, das Netzwerk zu nutzen.

Beispiel:

```
SERVER: Palm Server 0003
SERVER: Getting necessary information...
SERVER: ---> Enter class name to load
Agent
SERVER: ---> Enter address to send to
localhost
SERVER: Agent to load: Agent
SERVER: Target host: localhost:40000
SERVER: Started in serving mode
SERVER: Loading class: Agent.class
AGENT: Hello from Constructor()
SERVER: Generating thread
SERVER: Running agent
AGENT: Hello from init()
AGENT: Hello from run()
```

[2]Java Archive

```
AGENT: Showing dialog
AGENT: OK-Event: Value = 67
AGENT: Closing dialog
AGENT: Dialog finished
AGENT: run() ended
SERVER: Returned from call
SERVER: Sending started... getting IP-Address
SERVER: 'localhost' resolved to 127.0.0.1
SERVER: Connecting
SERVER: Sending
SERVER: Closing socket
SERVER: finished...
SERVER: main() ended
```

4.2 Server im Empfangsmodus

Der Programmteil Listen ist eine Empfängereinheit, die nach dem Start auf Socketverbindungen auf einer bestimmten Portnummer[3] wartet. Erfolgt eine Verbindung, so empfängt er die dort ankommenden Daten, entpackt diese und deserialisiert die empfangene Klasse. Anschließend startet er einen neuen Thread, in dem eine bestimmte Agentenmethode gestartet wird und somit wird der Agent wiederum zur Ausführung gebracht. Gleichzeitig erwartet der Listeningserver wieder auf eine Verbindung auf der oben angegebenen Portnummer, um weitere Agenten entgegenzunehmen.

In diesem Teil der Umgebung werden alle bereits im Abschnitt 4.1 genannten Komponenten benötigt: *Dynamisches Laden von Klassen*, *Objektserialisierung*, *Netzwerkunterstützung* und *Unterstützung des ZIP-Formats*. Über deren Funktion und Wichtigkeit wurde an genannter Stelle bereits eingegangen. Desweiteren wird hier die Multithreadingfähigkeit der Virtual Machine beansprucht. Sobald eine Verbindung zustande kam und der Agent empfangen, entpackt und deserialisiert wurde, wird der Agent in einem eigenen Thread gestartet. Anschließend werden wie-

[3] hier: 40000

der ankommende Socketverbindungen entgegengenommen. Gleichzeitg wird der Agent ausgeführt. Dank Multithreading können so mehrere Agenten empfangen und ausgeführt werden. Es muß nicht erst der zuletzt empfangene Agent terminieren, um den nächsten entgegenzunehmen. Dies ist eine der wichtigsten Eigenschaften, die ein Agentensystem fordert. Ein Agentensystem, welches immer nur einen Agenten bearbeiten kann, wäre sehr einfach angreifbar. So müßte nur ein Agente migrieren, der nicht terminiert. Der gesamte Agentenserver wäre somit lahmgelegt.

Beispiel:

```
LISTEN: Palm Listen 0006
SERVER: No parameter given... Starting listen mode
LISTEN: Waiting for connection
LISTEN: Connect from localhost/127.0.0.1
LISTEN: Reading data
LISTEN: Closing sockets
LISTEN: Showing agent
LISTEN: Generating thread
LISTEN: Waiting for connection
AGENT: Hello from init()
AGENT: Hello from run()
AGENT: Showing dialog
AGENT: OK-Event: Value = 67
AGENT: Closing dialog
AGENT: Dialog finished
AGENT: run() ended
```

Die Klasse Listen entstand aus der ursprünglich entwickelten Klasse Server[4]. Sie kann daher durch entsprechend parametrisierten Aufruf auch weiterhin die Rolle des Senders übernehmen. Ein entsprechender Aufruf ist *java Listen Agent localhost 40000*. Dies hat dieselbe Funktion, wie in dem Beispiel Server angegeben.

[4]Ursprünglich war nur eine Klasse in der Rolle des Sendens und Empfangens gedacht, die in die jeweiligen Rollen durch entsprechend parameterisierte Aufrufe gewählt wurde. Durch die jedoch recht unkomfortablen Aufrufmöglichkeiten mancher Virtual Machines wurde die neue Server-Klasse abgespaltet und als eigenständige Einheit weiterentwickelt.

4.3 Agent

Die Agentenklasse besteht aus einer einfachen Eingabemaske. Es liegen zwei Versionen der Agentenklasse vor, einmal mit GUI-Komponente und einmal als reine textuelle Eingabeaufforderung. Die GUI-Komponente wird dazu benutzt die graphischen Fähigkeiten der Virtual Machines zu testen. Die textuelle Variante wurde entwickelt, um sicherzustellen, daß grundsätzlich der Betrieb möglich gewesen wäre, sollte das Zielsystem keine GUI-Komponenten zulassen oder damit Schwierigkeiten haben. Eine GUI-Komponente ist für den grundsätzlichen Betrieb der SeMoA-Plattform nicht zwingend erforderlich. Es gibt jedoch Agenten, die sie benutzen.

Diese Eingabemaske soll einen zu übertragenden Wert abfragen, der dann nach der Deserialisierung auf der Listen-Seite erscheinen soll. Da die *Class*-Datei nicht mit übertragen wird, muß diese auf beiden Seiten, dem Server und dem Listeningserver, vorhanden sein[5].

GUI-Komponenten in Java laufen immer in einem eigenständigem Thread. Auch hier ist die Multithreadingunterstüzung der Virtual Machine gefragt. So muß die GUI weiterhin fehlerfrei arbeiten, während andere Threads auf beispielsweise Netzwerkschnittstellen zugreifen oder auf Verbindung warten. Dieser Teil der Testumgebung dient hauptsächlich als Datenkontainer. Er repräsentieret einen Agenten, der zu einem System migriert und dort seine Daten präsentiert, die er anderweitig erhalten hat.

Konsolenausgaben des Agenten, sind bereits in den Beispielen in den Abschnitten 4.1 und 4.2 zu sehen. Abbildung 4.1 zeigt, wie sich der Agent mit GUI-Komponente präsentiert.

[5] SeMoA hingegen prüft, ob die vorhandenen Class-Dateien den geforderten entsprechen.

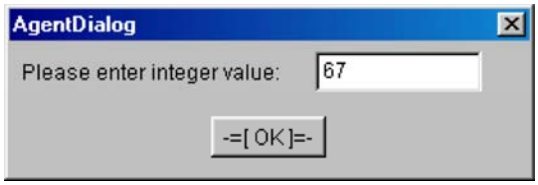

Abbildung 4.1: GUI-Komponente des Agenten

4.4 Zusammenfassung

Diese Testeinheit erlaubt es gezielt, grundsätzliche Funktionen der Virtual zu prüfen. Stellt sich heraus, daß die Testeinheit auf der Virtual Machine zufriedenstellend läuft, so kann über eine mögliche Portierung einer Agentenplattform für die Virtual Machine nachgedacht werden. Andernfalls wird die Realisierung einer Agentenplattform unmöglich, ohne an der Virtual Machine selbst Veränderungen vorzunehmen. Die fehlenden Eigenschaften mütßten dann erst in der Virtual Machine implementiert werden.

Teil II

Palm Vx

Kapitel 5

Palm Vx - Das Gerät

In diesem Kapitel wird auf den Palm[1] Vx (Abbildung 5.1) als Gerät eingegangen. Seine technischen Eigenschaften, sowie auch seine Erweiterbarkeit werden kurz umrissen. Außerdem wird seine Erfolgsgeschichte und der Grund, warum dieses Gerät in Betracht gezogen wurde geschildert. Abschliessend wird ein Ausblick gegeben, was die Zukunft dem Palm Vx möglicherweise bringt.

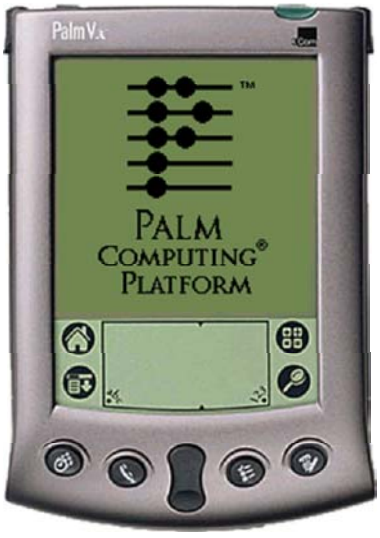

Abbildung 5.1: Palm Vx

[1] engl.: Handfläche

5.1 Technische Details

Der Palm Vx, der große Bruder des Palm V, ist ein weiteres Modell der erfolgreichen Palm-Reihe. Obwohl der eigentliche Hersteller *Palm Inc.* ca. 1 Jahr nach der Einführung der ersten *PalmPilot*-Modelle im März 1996 von 3Com aufgekauft wurde, führt Palm Inc. weiterhin die Entwicklung und Herstellung der Palms unter altem Namen fort.

Eine Auflistung der technischen Eigenschaften des Palm Vx zeigt Tabelle 5.1. Seine Ausstattung erlaubt es dem Palm Vx von Haus aus, ca. 10000 Adressen, Termine für ca. 5 Jahre (3000 Einträge), 3000 „To-Do"-Einträge, 3000 Memos und 400 Emails zu speichern, sowie eine Reihe von Zusatzprogrammen. Die Eingabe dieser Daten erfolgt dabei entweder mit Hilfe der mitgelieferten Software *Palm Desktop* via *HotSync*, direkt auf dem Palm mit Hilfe des Stifts über Schrifterkennung oder eingeblendeter Tastatur, oder von anderen Geräten via Infrarotschnittstelle. Diese erlaubt es auch, in Kombination beispielsweise mit einem kompatiblen Handy, Zugang zum Internet oder anderen Onlinediensten zu ermöglichen. *HotSync* ist eine Anwendung, die Daten zwischen DesktopPC und Palm synchronisiert. Über sie werden die Datenbanken, sowohl auf dem Palm, als auch auf dem DesktopPC, aktualisiert und zu installierende Programme übertragen.

5.2 Erweiterungsmöglichkeiten

Der Palm erfreut sich seit seiner Einführung großer Beliebtheit, wodurch auch viele Entwickler eine Chance sehen, ein gutes Gerät noch besser und vielseitiger zu gestalten. So wurden eine Reihe von Zusatzprogrammen und Modulen entwickelt, die den Palm zu einer modernen Kommunikationseinheit und mobilem Helfer in fast allen Lebenslagen machen. Es existieren bspw. Wireless-LAN Module, Modems, GSM-Module, GPS-Empfänger und aufklappbare externe Tastaturen. Aber nicht nur Seitens der Hardware gibt es diese Erweiterungsmöglichkeiten, die durch seine offene Hard- und Softwarestruktur erst ermöglicht werden. Es existiert auch Zusatzprogramme, die verschiedene Aufgaben übernimmt, die dem Palm bei seiner Entwicklung nicht zugedacht waren. Hier gibt es zum Beispiel Software, die Eingaben durch Wortergänzung erleichtert, Routenplaner, die in Verbindung mit

Produkt	
Hersteller	Palm Inc. (Tochterfirma von 3Com)
Produktname	Palm Vx
Betriebssystem	PalmOS 3.1, PalmOS 3.5 (updatefähig)
Kommunikationsprotokolle	TCP/IP-Unterstützung
Schrifterkennung	Graffiti (Schrifterkennungssoftware)
Hardware	
CPU	Motorola Dragonball EZ, 20 MHz
RAM	8 MByte
RAM erweiterbar	nein
ROM	2 MByte
ROM oder Flash erweiterbar	nein
PCMCIA	nein
Compact Flash	nein
Laufwerke und andere Speicher	keine
Bildschirm	
Auflösung	160x160 Pixel
Bildschirmtyp	Liquid Crystal Display
Anzahl Farben	16 Graustufen
Video Ausgang	nein
Verbindungsmöglichkeiten	
Kommunikationsschnittstellen	Modem (optional), Seriell, IrDa
Docking Station	ja
Verschiedenes	
Eingabemöglichkeiten	Stift, externe Tastatur (optional)
Audio	nein
Maße	7,8 cm x 11,4 cm x 1 cm
Gewicht	113 g (mit Batterie)
Stromversorgung	Lithium Ionen Akku

Tabelle 5.1: Technische Details — Palm Vx [51]

dem externen GPS-Modul den Weg weisen, eBook-Reader und vieles mehr. Einige Stärken und Schwächen des Palm Vx listet Tabelle 5.2 auf.

+	Größe
+	Gewicht
+	Preis
+	LI-Ion Akku
+	Lange Batterielaufzeit
+	Design für Links- und Rechtshänder geeignet
+	Einstellbare Hintergrundbeleuchtung
-	Geschwindigkeit
-	Ressourcen
-	Erweiterbarkeit
-	Inkompatibel zu anderen Palm Modellen
-	Keine Multimediafähigkeit
-	Display je nach Licht schwer lesbar
-	Graustufen

Tabelle 5.2: Stärken und Schwächen — Palm Vx

Auch Java (und einige Derivate, z.B. Waba [71]) ist auf dem Palm möglich. So stellen verschiedene Hersteller Virtual Machines bereit, um auch auf der PalmOS-Plattform Java zu etablieren. Inwieweit diese Virtual Machines für das Projekt SeMoA und mobilen Agenten geeignet sind, soll im Kapitel 6 geklärt werden.

Der Palm wurde als potentielle Plattform ausgewählt, da er und die kompatiblen Konkurrenzprodukte einen sehr hohen Verbreitungsgrad haben. So würde eine Palm-Portation von SeMoA das Interesse einer großen Zahl von Anwendern wecken.

5.3 Die Zukunft des Palm

In seinen Anfängen sprachen allein die, mittlerweile rückläufigen, Verkaufszahlen der verschiedenen Palm-Modelle für sich. Der Rückgang läßt sich einerseits an der größer werdenden Konkurrenz, die Palm-Klone bzw. eigene Entwicklungen

auf den Markt gebracht haben, erklären. Andererseits zeigt der Palm im Bereich *Multimedia*, ein moderner werdender Trend, nicht seine Stärken. Der Palm wird wohl noch einige Zeit seine Fangemeinde haben und es wird auch in naher Zukunft Erweiterungen und neue Software für ihn geben, jedoch früher oder später wird er durch ein anderes Produkt ersetzt werden.

Kapitel 6

Virtual Machines für Palm

In diesem Kapitel sollen einige der zur Verfügung stehenden Virtual Machines, die für den Palm Vx in Frage kommen, vorgestellt werden. Eine Liste verfügbarer Virtual Machines, nicht nur für PalmOS, findet sich bei JavaMobiles [25].

6.1 Sun Java2 Platform Micro Edition

Sun Microsystems, die Entwickler von Java, bieten eine eigene Virtual Machine für mobile Geräte an. Sie haben entschieden, daß es nicht eine Virtual Machine für verschiedene „Größen“ von Geräten geben kann. Abbildung 6.1 zeigt die drei erhältlichen Varianten der Java2 Plattform und zusätzlich die Java Card Api. Die Standard Edition enthält alle nötigen Tools und Klassen, die für die eigene Entwicklung und Betrieb von Java Applikationen benötigt werden. Die Enterprise Edition enthält zusätzlich noch *Enterprise JavaBeans*, die Entwicklern eine Reihe von Lösungen und Lösungsansätzen zu komplexen Problemen bereitstellen. Die Micro Edition stellt eine Virtual Machine für auf mobile Endgeräte und eingebettete Systeme bereit.

Die Java2 Platform Micro Edition (kurz JavaME) existiert in zwei Variationen, einmal dem CDC[1]-Profil und der CVM. Die zweite Variante, die für den Palm Vx gültige, ist die Zusammensetzung aus CLDC[2]-Profil und der KVM.

[1] Connected Device Configuration

[2] Connected Limited Device Configuration

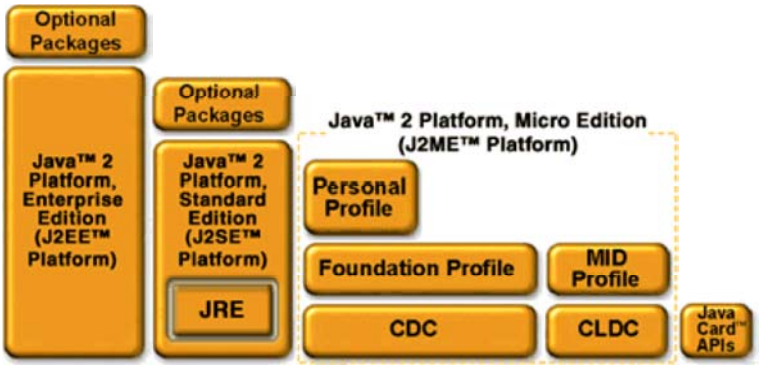

Abbildung 6.1: Aufteilung der Java Virtual Machines von Sun Microsystems [63]

Connected Limited Device Configuration

Um JavaME auf möglichst vielen Geräten einsetzen zu können, wurden zwei Konfigurationen beschrieben. CLDC definiert eine Java Plattform für kleine, ressourceneingeschränkte Geräte, die typischerweise einen Speicherausbau von 160 bis 512 kByte haben. Die Konfiguration besteht aus der KVM[3] (Abbildung 6.2 und Basisklassen, die jedes Device dieser Konfiguration unterstützt. Diese Klassen bestehen zum großen Teil aus den Grundklassen der Jave2SE und teilweise aus Klassen, die speziell für diese Geräte zugeschnitten wurden, dabei aber nicht Teil von Java2SE sind. Die zugrundeliegende Virtual Machine wurde speziell für diese Art von Geräten entwickelt.

Folgende Bereiche werden von CLDC abgedeckt:

- Basisbibliotheken von Java (`java.lang.*`, `java.util.*`)
- Ein-/Ausgabe
- Netzwerkunterstützung
- Sicherheit — Klassen werden auf Kompatiblität vor dem Ausführen überprüft, Anwendungen laufen in einer „Sandbox“, geschützte Klassen in Systempaketen können von Anwendungen nicht überschrieben werden.
- Internationalisierung

[3] Kilo Virtual Machine

Abbildung 6.2: KVM (Java2ME) auf dem Palm Vx

Durch CLDC werden nicht abgedeckt:

- Life-Cycle Management — Installation, Starten/Ausführen, Löschen
- Benutzerschnittstelle
- Ereignisbehandlung

CLDC beinhaltet weitestgehend die Spezifikationen der Sprache Java, wie sie GOSLING, JOY und STEELE [14] beschreiben. Ebenso werden, bis auf später genannte Punkte, die Spezifikationen der Virtual Machine, wie sie von LINDHOLM und YELLIN [38] beschrieben werden, umgesetzt. Nicht unterstützt werden:

- Fließkommazahlen (Float und Double),
- die Methode `Object.finalize()`,

- die meisten Subklassen von `java.lang.Error`,
- JNI - Java Native Interface.
- benutzerdefinierte Class-Loader.
- Reflexionseigenschaften,
- Thread Groups oder Daemon Threads,
- Weak References und
- Fehlerbehandlung nur stark eingeschränkt.

Gründe hierfür sind strikte Speicherbegrenzungen und potentielle Sicherheitsprobleme, die durch Fehlen der vollen J2SE Sicherheit entstehen. Fließkommazahlen wurden aufgrund fehlender Hardwareunterstützung auf der anvisierten Klasse von Geräten entfernt.

Welche Klassen im einzelnen die CLDC umsetzt, kann in *Java2 Platform Micro Edition Technology for Creating Mobile Devices, White Paper* nachgelesen werden. [66]

Als Erweiterung dieser Konfiguration zählt das MIDP — Mobile Information Device Profile.

> „MIDP is the first profile available for the J2ME mobile design center. The combination of CLDC and MIDP provides a complete environment for creating applications on cell phones and two-way-pagers.“ [68]

Connected Device Configuration

Im Unterschied zur CLDC beinhaltet die CDC die vollwertige Java2 Platform Virtual Machine (CVM[4]). Dazu kommen minimalisierte Klassenbibliotheken und

[4] Classic Virtual Machine

APIs. Als Zielplattformen visiert CDC Geräte mit 2 MByte Speicher, darunter sowohl RAM, ROM oder Flashspeicher. CDC enthält alle in CLDC definierten Klassen. Abbildung 6.3 zeigt die Beziehung von CLDC, CDC und J2SE Klassen. Klassen, die außerhalb von J2SE liegen, dürfen den Namen `java.*` nicht benutzen. [67]

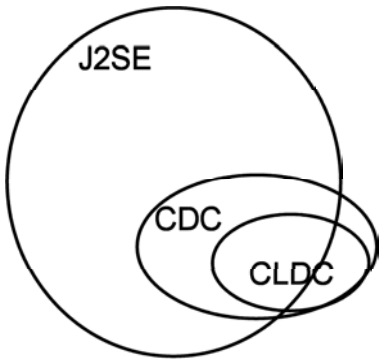

Abbildung 6.3: Beziehung von J2SE, CDC und CDLC

Das Basis Paket für CDC enthält folgende Java-Klassen:

- `java.lang`
- `java.util`
- `java.net` — UDP Datagramm und Datei Ein-/Ausgabe
- `java.io` — Datei Ein-/Ausgabe
- `java.text` — Minimale Unterstützung für I18n[5] (z.B. Fehlermeldungen)
- `java.security` — Minimale Sicherheit und Verschlüsselung für Objektserialisierung

Eigenschaften der KVM

Werden die Eigenschaften der KVM und die Forderungen, die die Portierung von SeMoA mit sich bringen, verglichen, so lassen sich bei der KVM bedeutende

[5] Abkürzung für „Internationalization“. Die Abkürzung besteht aus dem Anfangsbuchstaben „I“, dem Endbuchstaben „n“ und der Anzahl Buchstaben dazwischen „18“

Mängel feststellen. Sie besitzt beispielsweise keinen JIT-Compiler, was bedeutet, daß alle Klassen, die via KVM gestartet werden sollen, bereits vorher in einen, auf dem Zielsystem verständlichen, Maschinencode gebracht werden müssen. Die KVM bietet zu diesem Zweck auf der Entwicklungsseite entsprechende Werkzeuge, die für den Palm typische *.PRC*-Dateien erzeugen. Viel schwerwiegender ist aber die Tatsache, daß die KVM kein Multithreading und damit auch keine Thread Groups an sich unterstützt. Damit ist es die Aufgabe des Programmierers, eine quasiparallele Abarbeitung einzelner Programmteile, falls erwünscht, zu simulieren. Für ein Mobiles Agentensystem stellt dies ein hohes Sicherheitsrisiko dar, da auf die Kooperation der Agenten (bzw. deren Programmierer) gebaut werden muß. Ein maliziöser Agent könnte das gesamte System blockieren. Die KVM bietet auch von sich aus keine AWT-Komponenten, sowie Unterstützung für die Fließkommaformate Float und Double. Diese können aber durch frei verfügbare Zusatzkompenten[6] integriert werden. Objektserialisierung und das dynamische Nachladen von Klassen, sowie ZIP-Unterstützung sind realisiert. Ein Garbage Collector ist zwar vorhanden, jedoch ist er sehr simpel konzipiert und nicht für eine längere Ausführung der Virtual Machine ausgelegt. Die KVM unterstützt weitgehend die Spezifikation der Java Version 1.2. Eine Übersicht der Eigenschaften, im Vergleich mit den anderen, untersuchten Virtual Machines, findet sich am Ende des Kapitels in Tabelle 6.1 auf Seite 48.

Es läßt sich erkennen, daß die KVM für eine Portierung von SeMoA ungeeignet ist, es sei denn, es würden eine Reihe von Erweiterungen und Veränderungen vorgenommen.

6.2 IBM Visual Age Micro Edition

Visual Age Micro Edition (oder kurz: VAME) von IBM , basiert auf demselben Hintergrund wie die Virtual Machine von Sun Microsystems. Die vorliegende Version von VAME bezieht sich auch auf das CLDC-Profil, sowie auf das MIDP[7], und bietet damit Java- Kompatibilität für viele Embedded Systems. VAME binhaltet die sogenannte J9 Virtual Machine und einige Zusatzwerkzeuge. Die J9 VM

[6] AWT-Komponenten mit kAWT / Floating und Double mit MathFP

[7] Mobile Information Device Profile

wurde, basierend auf den JDK Version 1.2.2 Spezifikationen entwickelt. Es wurde dabei berücksichtig, daß Zielgeräte mitunter limitierte Ressourcen aufweisen können.

VAME existiert für eine ganze Reihe von Plattformen, beispielsweise AIX/PPC, ITRON/SH4, Linux/ARM, Linux/PPC, Linux/X86, PalmOS/68K, PocketPC/ARM, PocketPC/MIPS, Solaris/SPARC, QNX/ARM, QNX/MIPS, QNX/PPC, QNX/SH4, QNX RTP/X86, QNX/X86 etc. [21]

Abbildung 6.4: IBM Visual Age Micro Edition (VAME) auf dem Palm Vx

Die Visual Age Micro Edition Virtual Machine erweist sich in ihren Eigenschaften als ein wenig fortschrittlicher als das Konkurrenzprodukt von Sun Microsystems. Sie besitzt aber, ebenso wie die KVM, keinen JIT-Compiler. Auch hier müssen die Klassen vorher in den entsprechenden Maschinencode übersetzt werden. Sie unterstüzt auch keine Thread Groups, aber wenigstens bietet sie Multithreading an. Im Unterschied zur KVM kann eine Klasse einfach weitere Threads erzeugen, die dann eigenständig laufen. Das Fehlen von Thread Groups macht es für SeMoA

schwierig, die Agenten zu überwachen. Es kann nicht festgestellt werden, wann ein Agent und alle von ihm erzeugten Threads, zur Migration bereit ist. Der Garbage Collector ist im Gegensatz zur KVM auch für Langzeitbetrieb vorgesehen.

> „J9's primary technique for management and reclamation of freed memory (garbage collection) is a scheduled incremental garbage collector that is configurable for time slice and maximum period of execution. The collector is precise, accurate, and compacting, avoiding memory leaks and fragmentation." [22]

Es stehen einige optionale Bibliotheken zur Verfügung, beispielsweise AWT und RMI. Auch unterstützt diese Virtual Machine bereits Java Version 1.2.2 und ist damit zu diesem Zeitpunkt die modernste für den Palm. Leider unterstützt diese Virtual Machine aber nicht die ZIP-Kompression, was aber keinen schwerwiegenden Mangel ausmacht. Auch Objektserialisierung gehört leider nicht zu ihrem Repertoire, was für den Einsatz von Mobilen Agenten eine essentielle Funktion darstellt. Somit ist auch diese Virtual Machine für SeMoA in derzeitiger Fassung nicht geeignet. Eine Liste der wichtigsten Eigenschaften ist in Tabelle 6.1 auf Seite 48 im Vergleich mit den anderen Virtual Machines zusammengetragen.

6.3 Kada Mobile KadaVM

Die von Kada Systems entwickelte Virtual Machine *KadaVM* (Abbildung 6.5) besticht in erster Hinsicht durch eine fast vollständige Liste von Features, die in ihr realisiert wurden und die sich in sonst keiner Virtual Machine finden. Realisiert wurden JIT-Compiler, Multithreading, AWT, Floating Point und Double Klassen, vollständiger Garbage Collector, Class Finalization, Exception Handling, Object Serialization, Dynamic Class Loader, JVM Debugger Interface, ZIP/JAR File support, Thread Groups, Localization, SQL-Klassen, Math-Klassen, Text-Klassen, Net-Klassen und mehr. Kada gibt ihrer Virtual Machine eine vollständige Kompatiblität zur Java Version 1.1.8[8]. Viele der Voraussetzungen, die SeMoA mit sich bringt, sind realisiert worden. Die KadaVM erscheint daher als eine geeignete Virtual Machine für den Palm. Am Ende des Kapitels findet sich eine Übersicht der

[8] Version 1.2 soll in neueren Versionen folgen.

wichtigsten Eigenschaften, im Vergleich mit den anderen Virtual Machines für den Palm (vgl. Tabelle 6.1, Seite 48).

Leider bestand die KadaVM in der getesteten Version (August 2001) den Test mit der Testumgebung nicht.

Abbildung 6.5: KadaVM auf Palm Vx

Als Testumgebung für Palm-Applikationen steht ein, von Palm selbst angebotener, Emulator (POSE) für verschiedene Betriebssysteme zur Verfügung. Kada selbst empfiehlt, Applikationen mit Hilfe des Emulators zu testen, da dies, auf Grund der langsameren Übertragungsgeschwindigkeiten des Palms mit dem PC, schneller gehe. Das Problem bei Emulatoren ist meist, daß sie oft nicht vollständig kompatibel zu den emulierten Geräten sind. Auch POSE hat dieses Problem in einigen Punkten, dies ist jedoch für den Betrieb der VM nicht maßgeblich. Es ist somit kein Problem, die Virtual Machine auf dem Emulator auszuführen[9].

[9]Kada Systems setzt Version 30A6 als Minimum vorraus.

Das Problem bei der Virtual Machine von Kada war jedoch, daß diese sich auf dem Emulator und dem echten Palm unterschiedlich verhalten hat. Es gibt Stellen innerhalb der Testumgebung, die auf dem Emulator kein Problem darstellen, jedoch auf dem Palm anders ablaufen. So legte beispielsweise die *Socket.accept()*-Methode, die auf ankommende Verbindungen wartet, sämtliche Threads lahm. Sobald diese Methode aufgerufen wird, stellt der Agent sich selbst nicht mehr dar, reagiert nicht auf Eingaben und ähnliches. Nicht einmal die Ausgaben auf der Konsole wurden vollständig dargestellt. Eine Verbindung wurde jedoch akzeptiert und Daten fehlerfrei entgegengenommen. Auf dem Emulator jedoch trat dieses Phänomen nicht auf. Dort wurde der Agent weiterhin ausgeführt und reagierte, so wie es gedacht war. Wurde auf das Multithreading auf der Empfangsseite verzichtet, so funktionierte diese auch auf dem Palm wie erwartet.

Ein anderes Problem, vermutlich auf Timing-Problemen basierend, trat direkt nach dem Start des Empfängers auf. Der Empfänger wartet als erstes auf eine Verbindung von außen über die Socketschnittstelle. Grundsätzlich löste der erste Aufruf von *accept()* eine Exception aus, die darauf hindeutet, daß die Schnittstelle nicht bereit ist. Ein zweiter, direkt anschließender Aufruf führte dann zum Erfolg. Auch dieses Verhalten war auf dem Emulator nicht zu finden.

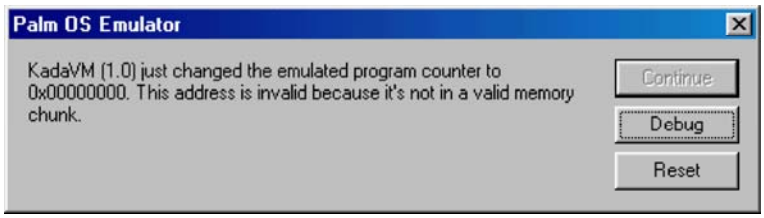

Abbildung 6.6: POSE-Fehlermeldung beim Serialisieren von Objekten (KadaVM)

Noch schwerwiegender ist ein Fehler, der beim Senden von Agenten, sowohl im Emulator als auch auf dem Palm, auftritt. So stürzen beide beim Aufruf der, im Sender nötigen, *writeObject()*-Methode mit unterschiedlichen Fehlermeldungen ab. Der Palm meldet einen „Unrecoverable Error“ und läßt sich nur via Reset wieder in Betrieb nehmen. Der Emulator meldet, wie in Abbildung 6.6 gezeigt, daß die Virtual Machine den Programmzähler auf eine ungültige Speicheradresse zu stellen versucht und dies nicht zulässig sei. Dies legt den Schluß nahe, daß auch

die Serialisierung von Objekten in der KadaVM noch nicht ausgereift ist. Aufgrund dieser Mängel ist die KadaVM ungeeignet für den Einsatz für Mobile Agentensysteme. Kada Systems hat sich bisher nicht zu den Mängeln geäußert. Es sind aber im Verlauf dieser Diplomarbeit bereits zwei neuere Versionen erschienen. Inwieweit diese die Anforderungen besser erfüllen, konnte aber nicht mehr untersucht werden.

6.4 Übersicht untersuchter Virtual Machines für den Palm

In diesem Abschnitt sollen tabellarisch die wichtigsten Eigenschaften der untersuchten Virtual Machines aufgelistet werden.

Feature	KVM	VAME	KadaVM
Operating System	PalmOS	u.a. PalmOS	PalmOS, WindowsCE
VM Footprint	40–80 kB	~300 kB	155 / 380 kB
VM Classes	min. 128 kB	100 kB–2 MB	50–460 kB
Memory Base	128 kB	k. Angabe	bis zu 6 MB
Memory Heap	32 kB	k. Angabe	bis zu 6 MB
AWT	mit kAWT	optional	✓
Exception Handling	limitiert	limitiert	✓
Floating Point	mit MathFP	×	✓
Double	mit MathFP	×	✓
Just-In-Time Compiler	×	×	✓
Multithreading	×	✓	✓
Thread Groups	×	×	✓
Garbage Collection	primitiv	✓	✓
Object Serialization	✓	×	✓
Dynamic Class Loading	✓	✓	✓
ZIP/JAR File	nur ZIP	×	✓
Java Security Layer	×	optional	×
Dynamic Proxy Generation	×	×	×
Java Version	1.2	1.2.2	1.1.8
URL	[64]	[21]	[28]
Preis	kostenlos	kostenlos	je nach Lizenzmodell

Tabelle 6.1: Eigenschaften der Virtual Machines für Palm

Kapitel 7

Mobile Agentenplattformen auf dem Palm

Hier wird kurz dargestellt, in wie weit andere Projekte Mobile Agentenplattformen für den Palm bereitstellen. Inwieweit eine Portierung von SeMoA auf dem Palm möglich ist, wird ebenfalls geschildert.

7.1 LEAP

LEAP[1] ist ein Projekt, mit dem Ziel eine FIPA[2]-konforme Agenten Plattform für mobile Endgeräte (PDA, Mobiltelefone, etc.) zu entwickeln. Ein weiteres Ziel von LEAP ist es, mit dem Agentensystem JADE[3] kompatibel zu sein. Tatsächlich benutzt LEAP die aktuelle JADE Plattform (Version 2.4) als Basis. JADE ist, ebenso wie SeMoA, komplett in Java realisiert, jedoch reicht hier bereits eine Java Version 1.2.

LEAP benutzt, um für möglichst jedes Gerät verfügbar zu sein, Profile. Das Grundprofil enthält nur die essentiellen Features, die FIPA fordert und eignet sich für die

[1] Lightweight Extensible Agent Platform

[2] Foundation for Intelligent Physical Agents - `www.fipa.org` - Die FIPA, bestehend aus etwa 60 Mitgliedsfirmen, erstellt Standards für Agenten/Agentensysteme.

[3] Java Agent DEvelopment Framework, vgl. [24]

kleinste angestrebte Zielplattform, z.B. ein Mobiltelefon. Das allumfassende Profil bietet Funktionen einer Agentenplattform, die für Desktopsysteme entwickelt wurde. LEAP wurde speziell für resourcenbegrenzte Geräte entwickelt. Es wurden nötige Interna von JADE neudesignt und neu implentiert um dieser Anforderung gerecht zu werden. Dabei wurden aber die APIs nicht verändert, so daß JADE-Agenten lauffähig bleiben. Je nach Profil müssen aber spezielle Agente verwendet werden. [1]

> „This remarkable feat has been achieved by adopting, adapting and improving parts of JADE, the Java Agent DEvelopment framework. More in details JADE-LEAP is a synergy of the LEAP libraries and the JADE platform." [35]

LEAP wurde erfolgreich auf J2SE-tauglichen Geräten, PalmOS (sowohl Palm als auch POSE-Emulator), iPAQ und anderen Geräten getestet. Die Homepage zeigt auch eine Abbildung, auf der ein Palm IIIc zu sehen ist, auf dem Leap mit Hilfe der Java2ME-Virtual Machine gestartet wurde, nachdem ein Agent eine ACL-Nachricht empfangen und ausgegeben hat.

7.2 SeMoA

Die hier untersuchten Java Virtual Machines zeigen deutliche Defizite, was die Voraussetzungen der SeMoA-Plattform betrifft. Um SeMoA auf einem Palm zu realisieren sind bedeutende Änderungen an der SeMoA-Plattform und Erweiterungen der Virtual Machines notwendig. Auf jeden Fall müssen Teile der Sicherheitsarchitektur von SeMoA verändert werden. Durch geeignetes Anpassen der Klassen und Schnittstellen wäre es vermutlich möglich, SeMoA in soweit zu reduzieren, daß die *Java2 Security* nicht weiterhin als Basis benutzt wird. Dadurch entstünden aber andere Sicherheitsrisiken, die maliziöse Agenten ausnutzen könnten. Problematisch ist auch die momentane Unterstützung des *Multithreading*. Hier muß eine fehlerfreie Unterstützung seitens der Virtual Machine gegeben sein. SeMoA dieser Eigenschaft zu berauben, ist keine Option, da sonst Mobile Agentenplattformen immer nur einen Agenten ausführen könnten. Hier kann auch wieder durch

einen maliziösen Agenten die Plattform lahmgelegt werden. *Multithreading* ist daher auch sicherheitstechnisch relevant. Ein weiterer Aspekt betrifft die verwendete Java Version. Mit der Umstellung auf Java2 hat sich die Threadunterstützung stark geändert und einige in der API stehende Methode wurden ersetzt. SeMoA benutzt diese geänderten Konstrukte. Die Virtual Machines für den iPAQ basieren teilweise noch auf den älteren Spezifikationen. Auch hier müßten Klassen angepaßt, ausgetauscht oder ergänzt werden. Agenten die auf SeMoA laufen, sehen ebenfalls die neuere Spezifikation als gegeben an. Agenten, die auf einem *Palm-SeMoA* laufen sollen, müßten ebenfalls speziell angepaßt werden. So muß ein Entwickler vorher bereits wissen, auf welchen Plattformen sein Agent zum Einsatz kommen wird. Bei speziellen Palm-Agenten mag dies noch sinnvoll erscheinen. Bei Agenten, die nicht speziell für den Palm vorliegen, können damit Probleme auftreten. Hier müßte die Plattform den Agenten auf Palm-Tauglichkeit prüfen. Erst wenn sichergestellt ist, daß der Agent ohne Probleme lauffähig ist, darf er gestartet werden. Dies ließe sich über geeignete Eingangsfilter bei der Migration prüfen.

Ein weiteres Problem stellt die Größe der Plattform dar. In der momentan vorliegenden Version beansprucht SeMoA etwa 2 MB freien Speicherplatz für die Klassen. Dies entspricht etwa einem Viertel des Speicherplatzes bei einem „leerem“ Palm Vx. Hinzu kommen noch die Virtual Machine und die Klassen, die diese benötigt. Ein wenig Platz kann durch Entfernen der Beispielklassen aus dem SeMoA-Archiv geschaffen werden.

Eine Realisierung ohne bessere Unterstüzung durch die Virtual Machine erscheint derzeit als nicht sinnvoll. Die resultierende Agentenplattform wäre zu angreifbar und bräuchte speziell zugeschnittene Agenten.

Teil III

Compaq iPAQ

Kapitel 8

iPAQ 3660 - Das Gerät

Dieses Kapitel stellt den Compaq iPAQ 3660 (Abbildung 8.1) als Gerät vor. Analog zu dem entsprechenden Kapitel beim Palm Vx, werden hier die technischen Eigenschaften, die Erweiterbarkeit, sowie ein Ausblick beschrieben.

Abbildung 8.1: Compaq iPAQ 3660

8.1 Technische Details

Der iPAQ ist ein multimedialer PDA, der modernen Ansprüchen genügt. Er bietet selbstverständlich auch die bei PDAs üblichen Features, wie das Verwalten von Terminen, Adressen, Aufgaben usw. Darüber hinaus läßt er sich dank Mikrofon, eingeschränkt durch geringen freien Speicherplatz, als Diktiergerät benutzen. Sein Lichtsensor regelt automatisch die Stärke der Hintergrundbeleuchtung und schont damit den Akku, was eine längere Laufzeit zur Folge hat. Auch hier bieten sich als Eingabemöglichkeiten der Daten ein Stift, eine eingeblendete Tastatur und es lassen sich die Daten aus den gängisten Verwaltungsprogrammen vom Desktop PC übertragen, z.B. Microsoft Outlook und Lotus Notes. Besondere Merkmale sind der eingebaute Lautsprecher und eine 3,5 mm Stereo Klinke Buchse, die den Anschluss eines Kopfhörers erlaubt. Eine Infrarot-Schnittstelle (IrDa-kompatibel) kann, wie beim Palm, zur Verbindung mit anderen Infrarotgeräten wie Notebook, Handy und anderen PDAs benutzt werden. Eine genaue Auflistung wichtiger Details läßt sich aus Tabelle 8.1 entnehmen.

Produkt	
Hersteller	Compaq
Produktname	iPAQ 3660
Betriebssystem	PocketPC (Microsoft WindowsCE 3.0)
Kommunikationsprotokolle	TCP/IP-Unterstützung
Schrifterkennung	MS Transcriber, Character Recognition
Hardware	
CPU	Intel StrongARM Prozessor, 206 MHz
RAM	64 MByte
RAM erweiterbar	via PCMCIA oder Compact Flash
ROM	16 MByte
ROM oder Flash erweiterbar	nein
PCMCIA	ja, über Erweiterungsmodul
Compact Flash	ja, über Erweiterungsmodul
Laufwerke und andere Speicher	via PCMCIA oder Compact Flash
Bildschirm	
Auflösung	200x320 Pixel
Bildschirmtyp	3.77 Zoll TFT LCD (Drucksensitiv)
Anzahl Farben	4096 Farben (12 Bit)
Video Ausgang	nein
Verbindungsmöglichkeiten	
Kommunikationsschnittstellen	IrDa, COM Port (für USB/Seriell)
Docking Station	ja
Verschiedenes	
Eingabemöglichkeiten	Stift, Tastaturtool, externe Tastatur
Audio	Mikrofon, Lautsprecher, 3,5 mm Klinke Stereo
Maße	8,33 cm x 12,97 cm x 1,57 cm
Gewicht	178,6 g (mit Batterie)
Stromversorgung	950 mAh Lithium Ionen Akku

Tabelle 8.1: Technische Details — Compaq iPAQ 3660 [50]

8.2 Erweiterungsmöglichkeiten

Der iPAQ ist kaum älter als 1,5 Jahre und dennoch gibt es unzählige Erweiterungsmodule. Möglich wird das durch PCMCIA- und Compact Flash-Karten, beispielsweise Netzwerkkarten aus Notebooks, zu verwenden, die in einen speziell dafür vorgesehenen, optionalen Einschub gesteckt werden. Eine Liste mit kompatiblen Geräten findet sich in der *Kompatibilitätsliste* von COMPAQ [6]. Der iPAQ bietet eine offene serielle Schnittstelle, die von verschiedenen Erweiterungskits, wie GPS Empfänger, GSM-Module und andere, benutzt werden kann. Eine sicherlich für Präsentationen ideale Erweiterung, bietet sich einmal als Hardware- und einmal als Softwarelösung. Sie erlaubt entweder den Anschluß eines Standard VGA Monitors der die Anzeige des Bildschimrinhalts des iPAQ auf einem DesktopPC. Eine Stärken- und Schwächenliste zeigt Tabelle 8.2.

+	LI-Ion Akku
+	Geschwindigkeit
+	Ressourcen
+	Erweiterbarkeit
+	Flexibilität
+	Multimedia tauglich
+	Automatische Hintergrundbeleuchtung
+	Kontrastreiches Display
+	Farbiges Display
-	Größe
-	Gewicht
-	Preis
-	Kurze Batterielaufzeit
-	Staub kann sich unter der Glasscheibe sammeln

Tabelle 8.2: Stärken und Schwächen — iPAQ

Eine andere Gruppe von Anwendern hat auch die Vorzüge des iPAQ erkannt, und entwickelte für diese Architektur Linux Derivate, die das vorinstallierte Windows-CE ersetzt. Auf eines dieser Derivate wird später noch näher eingegangen, da unter

ihm die vielversprechendste Virtual Machine existiert.

8.3 Die Zukunft des iPAQ

Die Möglichkeit, ein anderes Betriebssystem zu benutzen und damit den iPAQ für spezielle Aufgaben adaptieren ist sicherlich eine der besten Eigenschaften, die der iPAQ mit sich bringt. Auch seine Erweiterbarkeit durch PCMCIA und Compact Flash spricht für ein zukunftstaugliches System. Bereits jetzt ließe sich der iPAQ mit einem GSM-Modul als Handy benutzen, da Mikrofon und Lautsprecher/Kopfhöreranschluss bereits verhanden sind.

Kapitel 9

Virtual Machines für iPAQ

In diesem Kapitel wird auf die verschiedenen Java Virtual Machines und ihre Fähigkeiten sowie Schwächen eingegangen. Dazu werden die verschiedenen Java Virtual Machines unter dem mitgeliefertem WindowsCE, sowie unter einem Linux Derivat betrachtet.

9.1 Java Virtual Machines für WindowsCE

Das auf dem iPAQ installierte WindowsCE 3.0 wird ohne Java Virtual Machine geliefert. Microsoft bietet selbst auch keine Virtual Machine an, obwohl sie sie im Desktopbereich in ihr Betriebssystem eingliedern.

9.1.1 Jeode

Die Firma Insignia Solutions wurde 1986 in Großbritannien gegründet. Die Firma sollte Lösungen entwickeln, um nicht-Intel-Plattformen die Ausführung von DOS und Windows-basierten Anwendung zu ermöglichen. Mit Hilfe der dadurch gewonnenen Erfahrung von PC Virtual Machines, spezialisierte sie sich auf Java kompatible Virtual Machines für ressourcenbeschränkte eingebettete Systeme. 1998 lieferte Insignia Solutions Betaversionen ihrer *Jeode Platform* und *Jeode Embedded Virtual Machine (EVM).* [61]

Mittlerweile ist die Jeode Platform für die verschiedensten Betriebssysteme verfügbar, unter anderem WindowsCE, Windows NT4, VxWorks, ITRON, pSOS, Nucleus, BSDi Unix und Linux. Als unterstützte Prozessorarchitekturen werden MIPS, ARM, Hitachi SH-3 und SH-4, PowerPC und Intel x86 genannt.

Jeode unterstützt voll die Spezifikationen die durch Suns Personal Java 1.2 und EmbeddedJava 1.0.3 festgesetzt wird. Das entspricht in etwa dem Funktionsumfang von JDK Version 1.1.8. Leider fehlen damit einige der benötigten Komponenten und somit läßt sich SeMoA momentan nur schwerlich für diese Virtual Machine portieren. Eine Übersicht der Eigenschaften findet sich in Tabelle 9.2 am Ende dieses Kapitels auf Seite 71.

Die Kriterien der Testumgebung bestand die Virtual Machine ohne Schwierigkeiten. Auf eine Portierung von SeMoA mit Hilfe dieser Virtual Machine wurde verzichtet, da es noch eine vielversprechendere Virtual Machine gibt, die eine Realisierung mit wesentlich geringerem Aufwand möglich machen kann (vgl Kapitel 9.2.2).

Um SeMoA auf Jeode zu realisieren sind bedeutende Änderungen in der Architektur von SeMoA notwendig. Vor allem die Sicherheitsarchitektur müßte komplett überarbeitet werden, da SeMoA auf die Sicherheitsarchitektur von Java2 aufbaut. Jeode unterstützt diese jedoch noch nicht. Auch beherrscht Jeode keine dynamischen Proxies. Dynamische Proxies vereinfachen Zugriffssteuerungen auf Methoden verschiedener Klassen erheblich. Da Jeode ansonsten alle wichtigen Anforderungen, die SeMoA stellt, beherrscht, sollte einer Realisierung bis auf die oben genannten Punkte, keine größeren Probleme bereiten.

9.1.2 KadaVM

Diese Virtual Machine kam, auf Grund der bereits beim Palm beobachteten Schwierigkeiten, nicht zum Einsatz. Es soll hier nur erwähnt werden, daß diese sowohl für PalmOS als auch für WindowsCE verfügbar ist (vgl Kapitel 6.3).

9.2 Java Virtual Machines unter LinuxARM

Das OpenSource Betriebsystem Linux ist auf Grund seiner Beschaffenheit und der Förderung entsprechender Interessentengruppen wohl auf nahezu jeder Plattform denkbar. Es haben sich verschiedene Linux Derivate für den iPAQ etabliert, die alle auf der Debian Portierung basieren. Unter den bekanntesten sind LISA mLinux, Handhelds.org Familiar und PocketLinux. Im Zuge dieser Diplomarbeit wurden LISA mLinux und anschließend Handhelds.org Familiar in den jeweils aktuellen Versionen[1] eingesetzt. Installationsanleitungen befinden sich in den Anhängen B.2 und C. Welche Vor- und Nachteile Linux auf dem iPAQhaben kann, zeigt Tabelle 9.1.

+	Zugriff auf Quellcode
+	Stabilität
+	Flexibilität
+	Alle typischen PIM-Programme enthalten/erhältlich
+	Treiber fuer exotische Geräte meist erhältlich
+	Remote Shells
-	WindowsCE-Programme laufen nicht mehr
-	Experten-Knowhow notwendig
-	Treiber fuer exotische Geräte werden benötigt
-	kaum Unterstützung seitens der Hersteller
-	Mehrbenutzerbetrieb momentan noch unausgereift

Tabelle 9.1: Stärken und Schwächen — LinuxARM

Besonders komfortabel sind Zugriffe auf den Linux-iPAQ durch *Remotes Shells*. Hier kommt bei beiden Varianten SSH[2] zum Einsatz. So konnten alle nötigen Konfigurations- und Installationsvorgänge bequem über Desktop-PC geschehen und mußten nicht mühsam mit Hilfe der Eingabemöglichkeiten des iPAQ erfolgen. Die dabei gebräuchlichsten Tools sind `gzip`, `tar` und `scp`. Es wäre auch möglich gewesen, einen FTP-Server und andere Internetdienste auf dem iPAQ zu starten.

[1] LISA mLinux 0.9, Handhelds.org Familier 0.4 und 0.5

[2] Secure Shell

9.2.1 Kaffe

Kaffe ist eine OpenSource Implementation einer Java Virtual Machine und ihrer Klassen. Das Projekt wurde von Tom Wilkinson gestartet und dank reger Unterstützung von Entwicklern aus der ganzen Welt realisiert. Kaffe ist derzeit in der Version 1.0.6 verfügbar und unterstützt das JDK 1.1.7. Einige Teile sind bereits JDK 1.2 bzw Java2 kompatibel. Darunter fallen *RMI*, *Java Class Loader*, `java.util`-Klassen, und `java.security`-Klassen. Die Eigenschaften findet sich aufgelistet in Tabelle 9.2 auf Seite 71 am Kapitelende.

Kaffe ist eine beliebte Virtual Machine in Zusammenhang mit den Linux Distributionen für den iPAQ und fand aus diesem Grund Einzug in die Liste der zur Verfügung stehenden Virtual Machines. Ihre bisherigen Defizite bezüglich Kompatiblität zu Java2, insbesondere Version 1.3.1, ließen auch diese Virtual Machine in den Schatten einer anderen Portierung treten, auf welche als nächstes eingegangen wird (vgl Kapitel 9.2.2. [72]

9.2.2 Blackdown JRE 1.3.1RC1

Die wohl derzeit fortschrittlichste Virtual Machine für PDAs liefert Blackdown, eine Gruppe, die sich auf das Portieren von Sun Microsystems' Java Virtual Machine auf andere Prozessoren und Betriebssysteme spezialisiert hat. In den Archiven finden sich unter anderem für den StrongARM Prozessor portierte Linux-Versionen der JVM 1.1.8 und 1.3.1, wobei letztere zum momentanen Zeitpunkt den Status *Release Candidate 1* innehat. Da Version 1.3.1 genau den Vorraussetzungen für SeMoA entspricht, wurde versucht, SeMoA mit Hilfe dieser Virtual Machine auf den zuvor genannten Linuxderivaten, mLinux und Familiar, auszuführen. Details finden sich am Ende des Kapitels in Tabelle 9.2 auf Seite 71. [3]

LISA mLinux 0.9

LISA systems AG ist eins der größten, unabhängigen Linux-Systemhäuser Norddeutschlands. Sie implementieren bieten Support für modernste Linuxtechnologien. Mit der Einführung von mLinux 0.6 im Winter 2000 hat das Unternehmen die

erste Linux Distribution für den iPAQ-Handhald von Compaq bereitgestellt und so eine Neupositionierung des Unternehmens in Richtung mobiler Businesslösungen für den Zukunftsmarkt UMTS vorgenommen.

Abbildung 9.1: LISA mLinux 0.9 auf dem iPAQ

Es stellt kein Problem dar, die Virtual Machine unter LISA mLinux 0.9 (Abbildung 9.1) zu installieren. Beim Betrieb stellte sich jedoch heraus, daß diverse Libraries fehlen, sobald versucht wird, ein GUI-Objekt (sowohl AWT, als auch Swing) zu erzeugen. Diese Libraries sind Teil des X-Servers und sind bestandteil des Betriebssystems. Ihr Fehlen ist daher kein Versäumnis der Virtual Machine. Dieses Linuxderivat wird ohne einen X-Server geliefert, was für die verwendete Oberfläche (QPE von Trolltech[3]) auch nicht benötigt wird. QPE stellt die üblichen Applikationen eines PDAs bereit. Für eine Java-Anwendung mit GUI-Komponenten ist dies aber leider nicht ausreichend. Auch ein Versuch, die fehlenden Libraries nachträglich zu installieren scheiterte mangels Flashspeicher. Die fehlenden Libraries wurden aus der Familiar-Distibution entnommen und nachinstalliert. Nachdem

[3] `http://www.trolltech.com/products/embedded/index.html`

alle fehlenden Bibliotheken ergänzt wurden, war sehr wenig Speicherplatz frei (unter 1 MB) und der Aufruf der Virtual Machine brachte den iPAQzum Absturz. Dies kann zum einen an inkompatiblitäten der ergänzten Bibliotheken liegen, da diese für eine andere Distribution bestimmt waren. Zum anderen wurden durch die ergänzten Bibliotheken kein vollwertiger X-Server installiert. Um mLinux einzusetzen müßte demnach ein funktionierender X-Server, mit den für die Distirbution passenden Bibliotheken installiert werden. Somit kann momentan mLinux nicht für SeMoA verwendet werden, zumindest nicht für Agenten mit GUI-Komponenten[4]. Ohne GUI-Agenten ergaben sich keine weiteren Problem mit der Testumgebung.

Handhelds.org Familiar 0.4 / 0.5

Das Familiar Projekt besteht aus einer Gruppe lose zusammenhängender Entwickler, die alle an der Entwicklung eines PDA-Betriebssystem der nächsten Generation teilnehmen. Momentan wird die meiste Entwicklungszeit dazu verwendet, ein stabile und vollwertige Linuxdistribution für Handheldcomputer der Serie Compaq iPAQ H3600. Desweiteren werden einige darauf aufsetzende Applikationen entwickelt. [17]

Zu den Haupteigenschaften der Familiar Distribution gehören:

- ein auf `XFree86/keithp` basierender X-Server, mit der aktuellsten Rendereinheit
- Unterstützung von True-Type Schriftarten in `rxvt`, `blackbox` und `fltk`
- `ssh` und `sshd` von OpenSSH
- JFFS2 Unterstützung — erlaubt Lese- und Schreibzugriffe auf das Flasheprom des iPAQ
- Integriertes Python v2.0 mit `PyGtk` und `PyGDKImlib`
- Kompatibel zu den Bibliotheken und Anwendungen von der Debian StrongARM Distribution. Die meisten Programme können ohne Schwierigkeiten direkt aus der Debian entnommen und gestartet werden.

[4]Tatsächlich ist der Betrieb von SeMoA ohne GUI bereits möglich, jedoch existiert noch ein Fehler in dieser Variante, die das korrekte Migrieren von bestimmten Agenten verhindert.

Dieses Linuxderivat läßt sich hervorragend durch sein Paketsystem (IPKG) konfigurieren und fehlende oder benötigte Komponenten installieren bzw. nicht benötigte deinstallieren. Für Familiar steht ein X-Server bereit und nach Installation der Java Virtual Machine funktionieren auch GUI-Komponenten, sowohl in der Testumgebung, sowie SeMoA selbst. Da Familiar ein echtes Linux ist, konnte die Installation von SeMoA, welches mit Ausnahme der Shellskripte auf Java basiert, direkt von einem Desktop-PC mit SuSE Linux 6.2 übernommen werden.

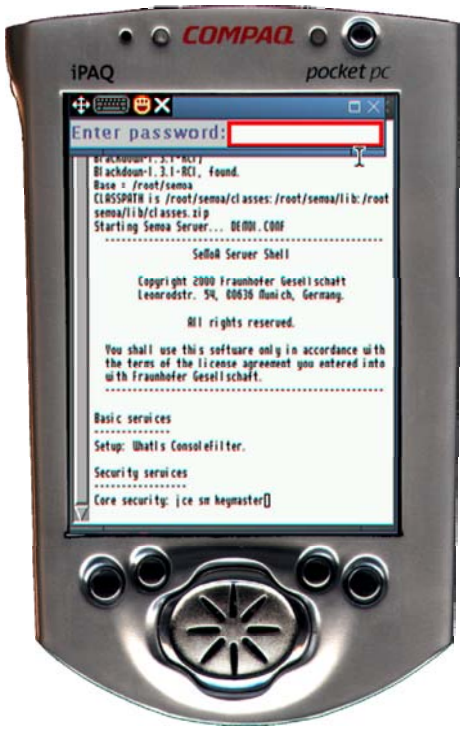

Abbildung 9.2: SeMoA auf iPAQ mit Familiar

Nachdem SeMoA erfolgreich über PPP mit einem Laptop oder Desktop-PC, auf Familiar 0.4 getestet wurde, sollte eine Anbindung an das Wireless LAN erfolgen. Hierzu wurde eine Cisco Aironet 4500 Karte zur Verfügung gestellt. Während der Kernel *2.4.3rmk2-np1* von Familiar 0.4 diese Karte nicht fehlerfrei unterstützte, wurde auch dieses Problem durch den Umstieg auf die neu erschienene Pre-Release 0.5 mit Kernelversion *2.4.7-rmk3-np1-devfs* behoben. Dort erfolgte die Anbindung ohne Probleme und nun liefen sowohl SeMoA, als auch die Wireless LAN-Karte wie beabsichtigt (siehe hierzu auch Abbilung 9.2).

Auch die im Januar erschienene Familiar 0.5 unterstützt die Karte fehlerfrei. Die ebenfalls erhältliche Version 0.5.1, sowie die für Februar angekündigte Version 0.6, konnten nicht mehr getestet werden.

Somit ließ sich SeMoA mit Hilfe von Familiar und Blackdowns JRE 1.3.1RC1 (vgl. Kapitel 9.2.2) auf einem Compaq iPAQ realisieren und als mobile Agentenplattform in ein bestehendes Netz einbinden.

9.3 Java als OS - SavaJe XE

Einen sehr interessanten Ansatz hält SavaJe Technologies bereit.

> „SavaJe Technologies' mission is to provide a powerful operating system based on Java2 Platform, Standard Edition (J2SE) technology, for next-generation information appliances and embedded devices — including smart phones, hand-held computers, personal digital assistants, set-top boxes, web pads and automotive internet access devices, and enterprise terminals." [59]

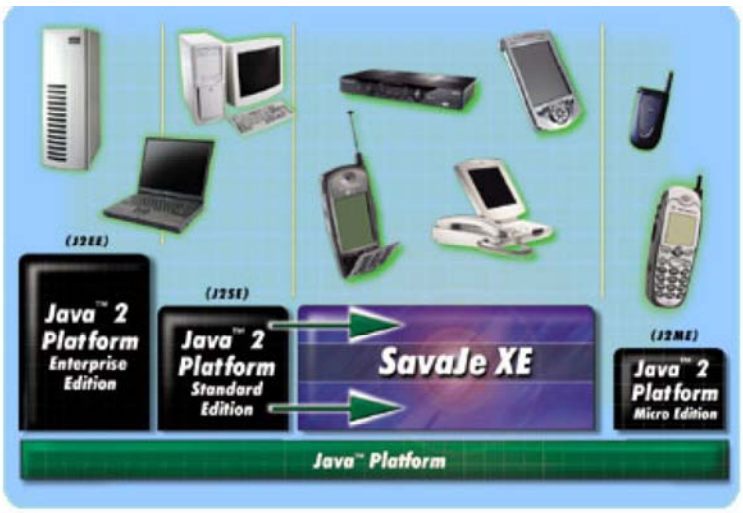

Abbildung 9.3: SavaJe XE - Anwendungsbereich

SavaJe XE versucht, wie in Abbildung 9.3 zu erkennen, die Lücke zwischen Desktop-PCs und Laptops, die ohne Probleme Java2 SE benutzen können und den Geräten, die Java2 ME als Java Plattform benutzen, zu schließen. SavaJe XE liefert mobilen Endgeräten der neueren Generation eine vollständige Implementierung einer Java2 Virtual Machine. Details zu den Fähigkeiten der Virtual Machine finden sich aufgelistet in Tabelle 9.2 auf Seite 71 im Vergleich zu den anderen untersuchten Virtual Machines.

Das entwickelte Betriebssystem bietet Eigenschaften, wie

- 32-Bit Multitasking und Multithreading Betriebssystem
- Integrierte Java Virtual Machine
- Advanced Power Management, geringer Speicherbedarf
- Unterstützung aller Java2 Standard Edition APIs, Pakete und Klassen einschliesslich Swing, Java 2D, AWT, volle Java2 Security, JDBC, Jini, RMI und CORBA
- Standardapplikationen wie Internet Browser, PIM, Email Programm, MP3 Spieler, Notizbuch und anderes
- Verfügbar für Compaq iPAQ H3600-Reihe und Psion NetBook. Versionen für Compaq iPAQ H3700-Reihe und H3800-Reihe angekündigt.

SavaJe XE benötigt etwa 12 MB FlashROM Speicher und benötigt minimal einen 190 MHz schnellen Intel StrongARM Prozessor (SA 1100/1110), sowie 32 MB RAM. Es steht eine Evaluationsversion zum freien Download bereit.

SavaJe XE basiert auf einer Familiar ähnlichen Linuxdistribution, startet direkt die Virtual Machine und präsentiert eine grafische Oberfläche. Alle gängigen Anwendungen wie Terminplaner, Adressverwaltung, Aufgabenliste stehen in der Evaluationsversion zur Verfügung. Zudem finden sich ein MP3-Spieler, ein Internet Browser und weitere Demonstrationsprogramme, die die Fähigkeiten der Java2 Virtual Machine zeigen sollen. Leider erwies sich die Eingabe mit dem Stift oder über Tastatur wegen einer fehlenden Kalibrierung als schwer. Auch eine Anbindung über serielle Schnittstelle oder PPP stellte sich als schwierig oder unmöglich heraus,

was aber an Beschränkungen in der Evaluationsversion liegen kann. Die vorliegende Version von SavaJe XE reagiert nicht auf Verbindungen über die serielle Schnittstelle. Auch die Liste der unterstützten Hardware ist begrenzt. So werden bisher nur folgende Ethernet-Netzwerkkarten unterstützt:

- 3Com PCMCIA Karten (auf 3C589X Basis, nur 10 MBit/s)
- Linksys Etherfast 10/100 PC Card (PCMPC100 v2)
- Linksys Etherfast 10/100 PC Card (PCM100 v2)
- Linksys Instant Wireless Network PC Card (WPC11)
- Lucent Orinoco (Gold und Silber)
- Xircom CompactCard (CF Karte) Ethernet10 (CFE-10)

Eine vollständige Liste unterstützter Hardware findet sich unter `http://www.savaje.com/products/supportedhardware.html`

Auf Grund des Zeitmangels war es leider nicht möglich, SavaJe XE, eine sicherlich interessante Alternative zu der Kombination von Familiar 0.5 und Blackdown JRE 1.3.1RC1, näher zu untersuchen.

9.4 Übersicht untersuchter Virtual Machines für den iPAQ

In diesem Abschnitt sollen tabellarisch die wichtigsten Eigenschaften der untersuchten Virtual Machines aufgelistet werden.

Feature	Jeode	Kaffe	Blackdown	SavaJe XE
Operating System	WindowsCE	Linux	Linux	Linux-basiert
AWT	✓	✓	✓	✓
Exception Handling	✓	✓	✓	✓
Floating Point	✓	✓	✓	✓
Double	✓	✓	✓	✓
Just-In-Time Compiler	✓	✓	✓	✓
Multithreading	✓	✓	✓	✓
Thread Groups	✓	k. Angabe	✓	✓
Garbage Collection	✓	einfach	✓	✓
Object Serialization	✓	✓	✓	✓
Dynamic Class Loading	✓	✓	✓	✓
ZIP/JAR File	✓	✓	✓	✓
Java Security Layer	×	✓	✓	✓
Dynamic Proxy Generation	×	×	✓	✓
Java Version	1.1.8	1.1.7	1.3.1	1.3.1
URL	[61]	[72]	[3]	[59]
Preis	$ 19.99	kostenlos	kostenlos	$ 100

Tabelle 9.2: Eigenschaften der Virtual Machines für iPAQ

Kapitel 10

Mobile Agentenplattformen auf dem iPAQ

In diesem Kapitel sollen kurz gängige, für den iPAQ verfügbare Mobile Agentenplattformen vorgestellt werden.

10.1 SeMoA

In diesem Abschnitt soll der Betrieb von SeMoA auf dem iPAQ vorgestellt und mit Hilfe der vorhandenen Beispielagenten demonstriert werden. Grundsätzlich unterscheidet sich der Betrieb durch, die von Java gegebene, Plattformunabhängigkeit nicht von dem auf anderen Plattformen. Einzelne Variationen sollten daher nur im Erscheinungsbild auftreten. Hier wird auch auf Auffälligkeiten hingewiesen, die evtl. durch Updates von SeMoA oder an entsprechend anderer Stelle behoben werden könnten.

10.1.1 Betrieb

SeMoA selbst ist komplett konsolengesteuert, wobei einzelne Agenten durchaus ihre eigene Oberfläche mitbringen können. Nach dem Start lädt SeMoA die in den Konfigurationsdateien eingetragenen Komponenten und präsentiert seine Eingabebereitschaft mit einem Prompt auf einer Shell. [58]

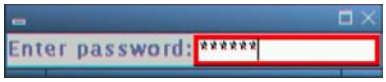

Abbildung 10.1: Passwortaufforderung von SeMoA auf dem iPAQ

Nach dem Start erscheint auf dem Display des iPAQ die Passworteingabeaufforderung (Abbildung 10.1) und nach Eingabe des Passworts[1] sich, entweder in dem Konsolenfenster, dem SSH-Terminal oder über Terminalprogramm an der seriellen Schnittstelle — je nachdem, wie SeMoA gestartet wurde, ein dem folgenden ähnliches Bild bieten sollte.

```
  ------------------------------------------------
                  SeMoA Server Shell

        Copyright 2000 Fraunhofer Gesellschaft
        Leonrodstr. 54, 80636 Munich, Germany.

                  All rights reserved.

  You shall use this software only in accordance with
  the terms of the license agreement you entered into
  with Fraunhofer Gesellschaft.
  ------------------------------------------------

Basic services
--------------
Setup: WhatIs ConsoleFilter.

Security services
-----------------
Core security: jce sm keymaster policy certfinder.
Bytecode filters: finalizer.
Agent filters: verify sign decrypt encrypt policy inform.

Miscellaneous services
----------------------
```

[1] Das Passwort lautet `volker`.

```
Registering lifecycles: default
Registering resources: server tmp.
Publishing event reflectors: agents restricted public.
Agent launcher filters: launcher import groups owner local.

Network services
----------------
Publishing gateways: ougate ingate raw.
Launching ingates: raw.
Network: vicinity httpd.
Atlas:.
Registering Servlets:.

Ready.
/:130>
```

Die Eingabe des Passworts erfolgt entweder über ein virtuelles Keyboard oder über die Schrifterkennung.

SeMoA ist nun einsatzbereit. Es können, wie sonst auch, mobile Agenten gestartet und empfangen werden. Um die Funktionalität zu prüfen kann beispielsweise der *JumpingAgent* genutzt werden. Dieser Agent springt zu einem gewählten Zielhost und wartet dort auf Eingabe eines neuen Ziels. Abbildung 10.2 zeigt den Jumping-Agent im Einsatz, wobei das eine Zielsystem der iPAQ und das andere ein Laptop ist, an.

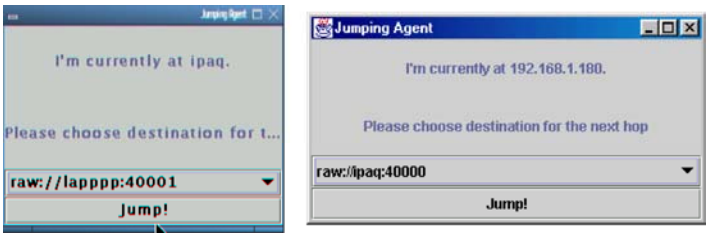

Abbildung 10.2: JumpingAgent auf iPAQ (links) und Windows-PC (rechts)

Um die Leistungsfähigkeit des iPAQzu zeigen, wird im folgenden ein Benchmark zwischen verschiedenen Stationen durchgeführt.

10.1.2 Benchmarks

In diesem Abschnitt soll ein Geschwindigkeitsvergleich zwischen Desktop-PCs und dem iPAQ beim Betrieb von SeMoA erstellt werden. Zum Einsatz kommen drei Personal Computer und der Compaq iPAQ 3660. Die wichtigsten Daten der benutzten Rechner zeigt Tabelle 10.1. Die Geschwindigkeitsmessung selbst erfolgt über den bei SeMoA mitgelieferten Benchmark-Agenten *BenchmarkAgent*. Der Agent wird so gestartet, daß er immer nur zwischen zwei Stationen hin- und herspringt. Es ist auch möglich, dem Agenten eine Liste von Zielsystemen zu übergeben. Der Agent springt dann jede Mögliche Route genau so oft an, wie ein weiterer Parameter angibt. Diese Variante wurde auf Grund der besseren Vergleichbarkeit nicht gewählt. Der verwendete Aufruf des Agenten lautet:
java DE.FhG.IGD.semoa.agent.BenchmarkAgent -rounds 20 -debug -route raw:// host1:port raw://host2:port. Für *host1:port* und *host2:port* wurden die entsprechenden Hostnamen bzw. Adressen und die Portnummer (40000 oder 40001, je nach Konfiguration) benutzt.

Es wurden drei Benchmark Tests durchgeführt, einmal zwischen zwei Hochleistungsrechnern, zwischen einer dieser Maschinen und einem Laptop und zwischen Laptop und iPAQ. Auf allen Rechnern läuft SeMoA unter der gleichen, für die jeweilige Plattform erhältlichen, Java-Version. Die einzelnen Filter von SeMoA wurden vor dem Benchmark initialisiert, um die Initialisierungszeit, welche nur beim ersten Migrieren auftritt, aus der Messung zu entfernen.

Arrakis	
CPU	AMD Thunderbird 1400 MHz
Betriebssystem	Microsoft Windows 98 (Second Edition)
Netzwerkanbindung	3com Fast Etherlink XL (10/100 MBit/s)
Netzwerkgeschwindigkeit	100 MBit/s
IP-Adresse	192.168.111.150
Giedi	
CPU	Intel Celeron2 600 MHz
Betriebssystem	SuSE Linux 6.2
Netzwerkanbindung	Fiberline Fast Ethernet Adapter
Netzwerkgeschwindigkeit	100 MBit/s
IP-Adresse	192.168.111.160
Laptop	
CPU	Intel Pentium MMX 233 MHz
Betriebssystem	Windows 98
Netzwerkanbindung	D-Link DFE-660
Netzwerkgeschwindigkeit	100 MBit/s
IP-Adresse	192.168.111.170 (LAN)
	192.168.1.171 (PPP seriell)
	143.140.13.57 (Wireless)
iPAQ	
CPU	Intel StrongARM 206 MHz
Betriebssystem	Familiar 0.5
Netzwerkanbindung	PPP via seriellem Kabel
	Cisco Aironet 4500 (Wireless)
Netzwerkgeschwindigkeit	115200 Baud
	11 MBit/s
IP-Adresse	192.168.1.180 (PPP seriell)
	143.140.8.139 (Wireless)

Tabelle 10.1: Technische Daten der am Benchmark beteiligten Rechner

Test 1 — Rechner „Arrakis“ und Rechner „Giedi“

Dieser Benchmark wurde ausgeführt, um die grundlegende Bearbeitungszeit von ein- und ausgehenden Agenten zu messen.

Aufruf: `java DE.FhG.IGD.semoa.agent.BenchmarkAgent -rounds 20 -debug -route raw://arrakis:40000 raw://giedi:40001`

Gesamtzeit	10050.0 ms
mittlere Migrationszeit	251 ms
Abweichung	46.0 ms

Tabelle 10.2: Benchmark-Ergebnis: „Arrakis“ und „Giedi“

Das Ergebnis des Test zeigt (Tabelle 10.2), daß die Gesamtzeit für die jeweils 20 Sprünge etwa 10 Sekunden beträgt. Arrakis beansprucht, da beide Maschinen ähnlich schnell arbeiten, etwa 5 Sekunden davon. Die mittlere Verweildauer des Agenten beträgt 251 ms. Die Standardabweichung von 46ms läßt sich mit geringem, aber vorhandenen, Netzwerkverkehr und Ausnutzung der CPU durch andere Tasks erklären.

Test 2 — Rechner „Arrakis“ und Rechner „Laptop“

Im zweiten Test soll die Bearbeitungsdauer des Laptops, der um einiges schwächer bestückt ist, gemessen werden.

Aufruf: `java DE.FhG.IGD.semoa.agent.BenchmarkAgent -rounds 20 -debug -route raw://arrakis:40000 raw://laptop:40001`

Gesamtzeit	27460.0 ms
mittlere Migrationszeit	687 ms
Abweichung	73.0 ms

Tabelle 10.3: Benchmark-Ergebnis: „Arrakis“ und „Laptop“

Das Testergenis läßt sich aus Tabelle 10.3 entnehmen. Hier fällt sofort die etwas höhere mittlere Bearbeitungszeit auf. Es läßt sich daraus ableiten, daß der Laptop

grundsätzlich etwa viermal so lange an einer Migration des Agenten arbeitet als Arrakis. Arrakis benötigt, wie im ersten Test ermittelt etwa 5 Sekunden für die Bearbeitung. Diese 5 Sekunden von den 27.4 Sekunden des zweiten Tests abgezogen ergeben eine ungefähre Bearbeitungszeit von 22 Sekunden für den Laptop, für 20 Migrationen, daß heißt etwa 1 Sekunde pro Migration. Auch hier zeigt die geringe Abweichung von 73 ms einen recht gleichmäßigen Verlauf des Tests.

Test 3 — Rechner „Laptop“ und iPAQ

Nachdem in etwa festgestellt wurde, welche Zeit der Laptop pro Migration benötigt, soll diese nun beim iPAQ festgestellt werden.

Aufruf: `java DE.FhG.IGD.semoa.agent.BenchmarkAgent -rounds 20 -debug -route raw://ipaq:40000 raw://laptop:40001`

Gesamtzeit	958510.0 ms
mittlere Migrationszeit	23963 ms
Abweichung	178.0 ms

Tabelle 10.4: Benchmark-Ergebnis: „Laptop“ und iPAQ

Der iPAQ, zumindest nominell in der Geschwindigkeitsklasse des Laptops, bremst die Migration des Agenten aus, wie der Tabelle 10.4 entnommen werden kann. Ausgehend von den Erkenntnissen aus Test 2, berechnet sich eine Bearbeitungsdauer des iPAQ von insgesamt 936 Sekunden oder 15,6 Minuten, eine recht hohe Zeit, jedoch noch im Bereich des Erträglichen. Daraus ergibt sich eine mittlere Bearbeitungsdauer von 45 Sekunden pro Migration. Die Anbindung an das Netzwerk betrug hierbei aber nur 115200 Baud. Auch hier zeigt die Standardabweichung von 178 ms, in Relation zu der Bearbeitungsdauer gesetzt, einen recht gleichmäßigen Verlauf des Tests. Ein ähnliches Ergebnis lieferte eine Messung mit der Wireless-LAN-Karte, was darauf hindeutet, daß die Übertragung des Agenten selbst, welcher ohne zusätzliche Last betrieben wurde, nicht ins Gewicht fällt.

10.1.3 Beispiel einer Anwendung

Mobile Anwendungen werden in Zukunft einen Grundstein der Kommunikation, Information und Unterhaltung bilden. Mit der Zunahme der Mobilität wächst auch das Kommunikationsbedürfnis und eine größere Vielzahl von Informationen werden benötigt. KAHMANN führt dazu weiter aus:

> „Für die Geschäftswelt geht es insbesondere darum, die Produktivität ihrer Arbeitsabläufe zu optimieren und flexibel agieren zu können, während für private Nutzer eher erhöhter Komfort, Flexibilität, Sicherheit, Zeitvertreib und Unterhaltung sowie ein bestimmter Life Style im Vordergrund stehen. Dazu wird der modernen Gesellschaft in Zukunft zu jedem Zeitpunkt und an jedem Ort eine Vielzahl an mobilen Kommunikations-, Informations- und Unterhaltungsapplikationen zur Verfügung gestellt, um ihre verschiedenen Bedürfnisse abzudecken." [29]

Diese Applikationen werden von KAHMANN in vier grundlegende Klassen eingeteilt und wie folgt beschrieben:

Kommunikationsdienste Diese Dienste erlauben die direkte und indirekte Interaktion zwischen Benutzern. Mit Benutzer werden in diesem Zusammenhang auch zunehmend Maschinen bezeichnet. Typische Beispiele für Kommunikationsdienste sind Sprache, E-Mail, SMS, Chat, Multimedia Messaging (z.B. Videopostkarten) und Telemetrie / Telematik (z.B. Fernsteuerung von Haushaltsgeräten, Navigation, Tracking - d.h. Aufzeichnung von gegenwärtigen Aufenthaltspunkten und Bewegungsmustern von Personen, Fahrzeugen, Objekten)

Informationsdienste Diese Dienste ermöglichen Benutzern Zugriff auf jegliche gewünschte Information zu jedem Zeitpunkt und an jedem Ort. Dabei können Informationen auf einen bestimmten Benutzerprofilen, Zeit und Ort zugeschnitten werden. Informationen können als Sprache, Text, Bilder oder Video übertragen werden und umfassen Nachrichten, Reiseinformationen, Auskunftsinformationen, Zugriff auf Firmendatenbanken, standortabhängige Werbung und ähnliches.

Unterhaltungsdienste Unterhaltungsdienste sind typische Freizeitdienste und umfassen interaktive Onlinespiele und -glücksspiele, Videosequenzen, auf Abruf verfügbare Musikstücke, das Herunterladen von Klingeltönen, Icons oder ähnliches.

Handels- und Transaktionsdienste Handels- und Transaktionsdienste gewähren dem Benutzer den jederzeitigen Zugriff auf Bank- und Brokeragedienste, kommerzielle Angebote des Einzelhandels sowie Auktionen.

Um die Möglichkeiten, die SeMoA in Verbindung mit dem iPAQbietet, wird ein Unterhaltungsdienst entwickelt. Die Anwendung stellt einen mobilen MP3-Player dar, der die MP3-Dateien vor dem Spielen von einem entferntem Rechner holt und dann abspielt. Eine Musiksammlung auf dem iPAQzu speichern, macht in der Standardausführung mangels Speicherplatz keinen Sinn.

Entwicklung

Zunächst wurde versucht, einen in Java geschriebenen MP3-Dekoder auf dem iPAQ zu betreiben. Dabei stellten sich zwei Schwierigkeiten sofort heraus.

Es gibt Schwierigkeiten mit dem JavaSound API. Die verwendete Virtual Machine sollte eigentlich JavaSound beherrschen. Grundsätzlich funktioniert die API auch, jedoch stellt sich bei der Ausgabe der Daten ein Problem ein. Die prinzipielle Soundausgabe mit JavaSound wird im folgenden kurz erläutert. Eine Anwendung schreibt Daten in eine *SourceDataLine*. Diese Leitung wird vorher über das *AudioSystem* angefordert und reserviert. Die Anwendung schreibt auszugebene Daten in diese Leitung, die in einem Puffer landen. Sobald der Puffer voll ist, wird die Anwendung solange blockiert, bis wieder Speicher im Puffer frei ist. Die Virtual Machine ist dafür zuständig, daß die im Puffer stehenden Daten abgearbeitet werden und auf der Soundkarte ausgegeben. Auf dem iPAQergab sich das Problem daß der Puffer nicht geleert wird, wohingegen die gleiche Anwendung auf anderen Systemen funktionstüchtig ist. Eine genauere Lokalisierung des Problems war nicht möglich. Auch eine Stellungsnahme von Blackdown liegt bisher noch nicht vor. Das eine Soundausgabe auf dem iPAQJava möglich ist, zeigten direkte Zugriffe auf das Sounddevice, welches unter Linux als `/dev/dsp` angesprochen werden kann. Somit ergab sich eine, wenn auch unschöne Lösung für dieses Problem. [45]

Schwerwiegender als die Soundausgabe wiegt die Geschwindigkeit des Java MP3-Dekoders. Als Decoder wurde *JavaLayer* von JAVAZOOM benutzt [26]. Die Dekodierung eines Frames dauert zu lange, sodaß ein „flüssiges" Abspielen unmöglich wird. Ein kurzer Blick in den Quellcode von *JavaLayer* läßt auch den Grund hierfür erkennen. Der JavaLayer-Dekoder arbeitet hautsächlich mit den Datentypen *Float* und *Double*. Beide Datentypen repräsentieren eine Fließkommazahl. Da der iPAQk- eine eigene Fließkommaarithmetik hat, muß diese rückführend auf Integer-Operationen simuliert werden. Dies beansprucht aber große Mengen an Ressourcen und Zeit. Das grundsätzlich ausreichend Kapazitäten für die Wiedergabe von MP3-Dateien vorhanden sind, demonstriert das Programm *Madplay*, welches auch als IPKG-Paket in der Familiar-Distribution erhältlich ist.

Madplay ist ein, auf Fixpunktarithmetik basierter, MP3-Dekoder. Als OpenSource-Projekt ist *Madplay* komplett in C/C++ geschrieben und liegt für verschiedene Plattformen vor. [36]

Der entwickelte MP3-Player stellt demnach eine SeMoA-Anbindung an *Madplay* dar.

Das Klassendiagramm (vgl. Abbildung 10.3) zeigt die Abhängigkeiten der entwickelten Klassen untereinander, sowie deren Eingliederung in vorhandene Klassen von SeMoA. Die einzelnen Klassen werden im Folgenden kurz vorgestellt:

MP3FrontEnd Diese Klasse dient als Starter für die GUI.

MainGUI Die Klasse `MainGUI` bildet die Schnittstelle zum Benutzer und konfiguriert die Agenten.

DirAgent Diese Klasse, abgeleitet von `RoundtripAgent` bildet den Agenten, der zu dem Zielsystem migriert und dort Unterverzeichnisse durchsucht.

DirBrowser Um die Unterverzeichnisse zu durchsuchen, kommt diese Klasse zum Einsatz.

MP3Path Diese Klasse stellt einen konfigurierbaren `<Default>`-Pfad bereit.

FileAgent Analog zu der Klasse `DirAgent` umfaßt diese Klasse den Agenten. Nachdem die Datei kopiert und der Agent zum Ursprungssystem zurückmigriert ist, wird der MP3-Player gestartet.

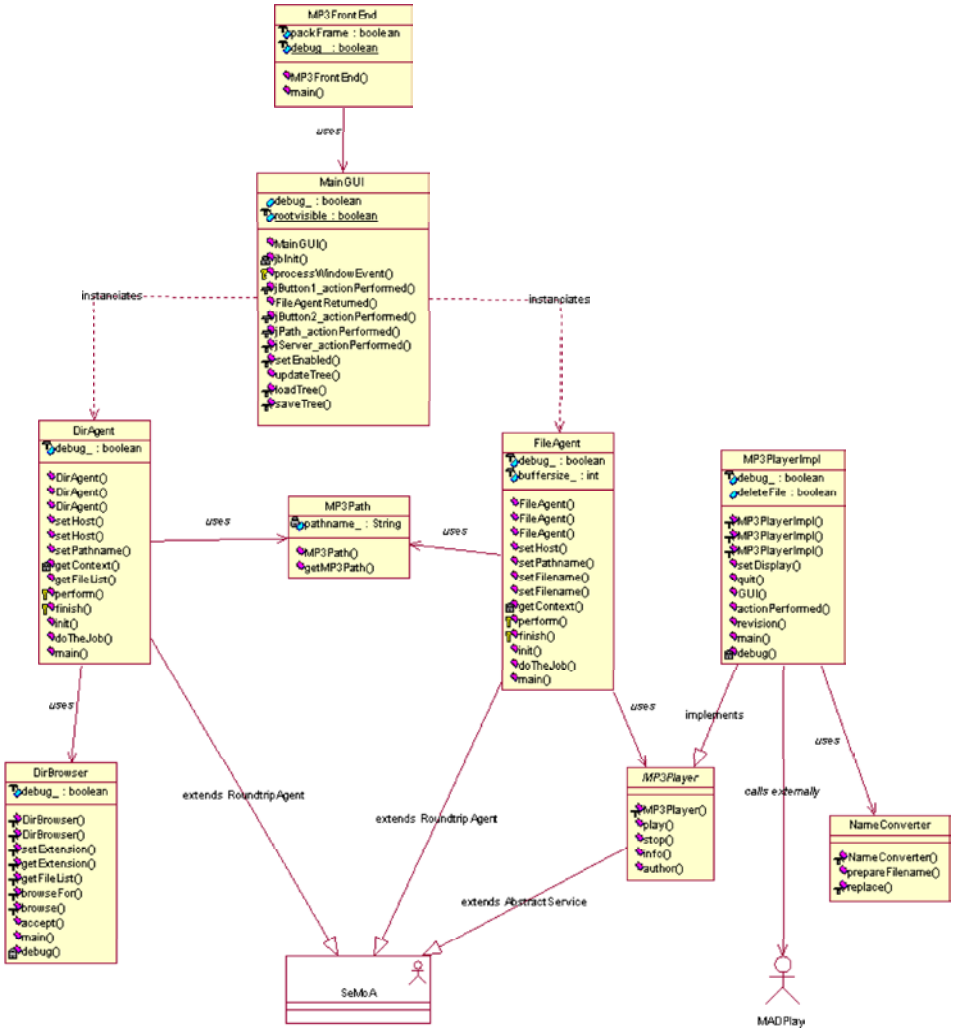

Abbildung 10.3: Klassendiagramm: MP3-Agent

MP3Player Dieses Interface stellt die Grundfunktionen eines MP3-Players zur Verfügung.

MP3PlayerImpl Eine mögliche Implementierung des `MP3-Player`-Interfaces. Hier wird extern das Programm `madplay` gestartet.

NameConverter Da sich im externen Befehl die Trennzeichen für Pfad und Dateinamen unterscheiden, wurde diese Klasse implementiert. Sie konvertiert in einen übergebenen Dateinamen entsprechende Zeichen.

Die mit `SeMoA` und `Madplay` bezeichneten Aktoren stellen die Schnittstellen zu den jeweiligen Programmpaketen dar.

Installation/Konfiguration

Eine spezielle Installation ist nicht nötig. Es müßen nur die Klassendateien der Virtual Machine zur Verfügung stehen, d.h. der `classpath` muß angepaßt werden.

Um einen reibungslosen Ablauf zu gewähren, muß SeMoA umkonfiguriert werden. Der MP3-Player z.B. Rechte für das Ausführen von Programmen haben, er muß Schreibrechte in dem temporären Verzeichnis haben usw. Um ihm diese Rechte zu geben sind folgende Permissions in der Datei `etc/semoa.policy` einzutragen:

```
BEGIN MP3
PERM java.io.FilePermission ./mp3/-  read
PERM java.io.FilePermission \\tmp    delete
PERM java.io.FilePermission \\tmp\\- read,write,delete
PERM java.io.FilePermission "<<ALL FILES>>" read,execute
END
```

Hinweis: Diese Notation wurde aus einer Konfigurationsdatei von SeMoA entnommen, die auf ein Windows-System zugeschnitten ist. Auch wenn Java sonst keine Unterscheidung zwischen \ und / macht, so ist dies hier dennoch von Bedeutung.

Die erste Zeile dieser Permission wird auch auf dem MP3-Server benötigt. Dort muß der Agent Leserechte für die MP3-Dateien bekommen. Es empfiehlt sich, wie hier dargestellt, diese Rechte in einem eigenen Block zusammenzufassen.

Um den MP3-Player zu starten muß, nur die Klasse `MP3FrontEnd` instanziiert werden. Um auch die Agenten zu nutzen, sollte der Start aus SeMoA erfolgen. Optional kann der Parameter `-debug` angegeben werden. Er veranlaßt die Anwendung ausführlichere Meldungen auf der Konsole auszugeben.

Betrieb

Der MP3-Player teilt sich in 3 Teile auf: eine steuernde GUI, ein Datei-Agent und ein Verzeichnis-Agent.

Der Datei-Agent (`FileAgent`) dient als Datenkontainer. Er migriert zu einem angegebenen Server und holt dort eine spezifizierte MP3-Datei. Anschließend kehrt er zu seinem Ursprungssystem zurück und startet die Wiedergabe der Datei.

Der Verzeichnis-Agent (`DirAgent`) migriert auch zu einem angegeben Server, jedoch durchsucht er ein angegebenes Verzeichnis nach MP3-Dateien und kehrt mit einer entsprechenden Liste zu seinem Ursprungssystem zurück.

Die GUI besitzt genau zu diesem Zweck zwei Funktionen: „Browse“ und „Fetch“.

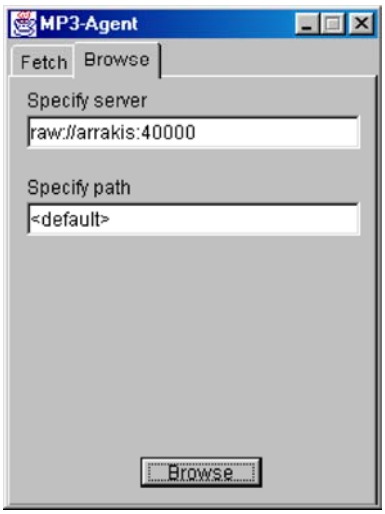

Abbildung 10.4: MP3-Player GUI: Browse Option

Unter *Browse* finden sich zwei Textfelder, in die die gewünschten Informationen eingetragen werden. Zum einen ist das der zu durchsuchende Server. Die Adresse wird in der in SeMoA üblichen Form angegeben. Zum anderen muß das Verzeichnis angegeben werden, in dem nach MP3-Dateien gesucht werden soll. Der Pfad `<default>` zeigt dabei auf einen auf dem jeweiligen Server gesetzten Standardpfad (derzeit `mp3/` im SeMoA-Installationsverzeichnis). Abbildung 10.4 zeigt Beispieleingaben.

Fetch hingegen zeigt eine Liste von Servern, dazugehörenden, bereits durchsuchten Verzeichnissen und die in den Verzeichnis liegenden Dateien. Hier kann die gewünschte Datei gewählt werden und sobald der Agent mit der Datei zurückgekehrt ist, wird diese von dem Agenten abgespielt. Die aktuelle Liste wird beim

Verlassen der GUI automatisch gespeichert. Beim nächsten Start der GUI steht somit wieder die Liste zur Verügung, sodaß nicht erst wieder Verzeichnisse durchsucht werden müßen. Abbildung 10.5 zeigt, wie die Datei *Tom Kubis Big Band - Naked Gun Theme.mp3* auf dem Server *raw://arrakis:40000* im Standardverzeichnis ausgewählt wurde.

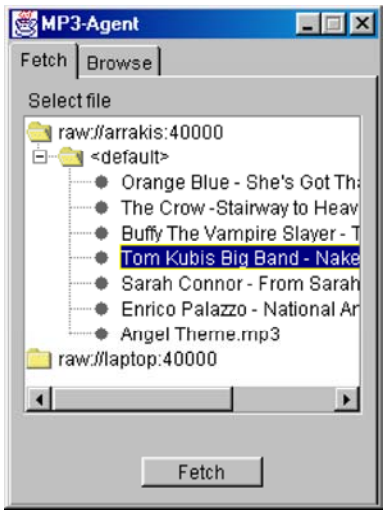

Abbildung 10.5: MP3-Player GUI: Fetch Option

Denkbare Erweiterungen

Als denkbare Erweiterung fällt sofort auf, daß der externe Player *Madplay* durch einen auf Java-basierten Player ersetzt werden kann.

Weiterhin wäre es möglich, den Verzeichnis-Agenten in folgenden Punkten zu erweitern:

- Rekursive Suche in Unterverzeichnissen — Momentan druchsucht der Agent nur das angegebene Verzeichnis und nicht darin liegende Unterverzeichnisse.
- Mehrere verschiedene Basisverzeichnisse — Der Agent könnte direkt mehrere Verzeichnisse als Basis für die Suche übergeben bekommen und in diesen nach MP3-Dateien suchen.

- Angabe von mehreren Hosts — Der Agent springt momentan zu einem Zielsystem, vollzieht dort die Suche und kehrt zurück. Denkbar wäre hier die Durchsuche aller im Netz gefundenen Server, oder einer Auswahl aus diesen.

Ähnliche Punkte wären auch im Datei-Agenten möglich:

- Übergabe einer Spielliste — Derzeit übergibt man dem Agenten einen Dateinamen und Server. Als Erweiterung ist die Übergabe einer Spielliste denkbar, in der mehrere Dateien (sogar von verschiedenen Servern) stehen. Der Agent könnte dann während die eine Datei gespielt wird bereits die nächste holen. Derzeit ist dies nur durch starten mehrer Datei-Agenten möglich. Allerdings muß selbständig koordiniert werden, daß nicht mehrere Dateien gleichzeitig abgespielt werden.
- Auswahl des Players — Derzeit wird fest das externe Programm *Madplay* aufgerufen, welches sich im Suchpfad befinden sollte. Hier könnte auf andere Player (beispielsweise WinAMP oder xmms) ausgewichen werden. Ein intern realisierter Player ist ebenso denkbar.

Je nach Erweiterung der Agenten, sollte die GUI ebenfalls entsprechend angepaßt werden.

10.2 LEAP

Auf die LEAP-Plattform wurde bereits in Kapitel 7.1 eingegangen. Aufgrund ihrer Struktur ist sie für jede Plattform denkbar, die mindestens J2ME-CLDC unterstützt. Abbildung 10.6 zeigt den Startbildschirm von LEAP, welches auf einem iPAQ gestartet wurde.

10.3 Grasshopper

Grasshopper [23] ist eine Agentenplattform, die von IKV++ GmbH 1998 das erste Mal vorgestellt wurde. Die Agenten Plattform Grasshopper 2, die neueste Version

Abbildung 10.6: LEAP auf dem iPAQ

der Plattform, bietet an, Java-basierte mobile Agenten zu entwickeln und zu betreiben. Auch hier werden die notwendigen Mechanismen zur Verfügung gestellt. Mit Hilfe eines Addons ist es möglich, die Plattform FIPA-konform zu erweitern.

Das Grundsystem erlaubt es, eine auf Grasshopper-basierende Umgebung bestehend aus Agenturen (Agentensystemen) zu erstellen. Die Agenturen werden in einer Domain, z.B. Intranet, gruppiert. Eine Agentur ermöglicht und kontrolliert die Ausführung, das Management, den Transport und die Kommunikation von Grasshopper-Agenten. Dabei werden folgende Komponenten/Dienste abgedeckt:

Sicherheit Grasshopper versucht den Host vor unauthorisiertem Zugriff durch Mobile Agenten zu schützen. Zum einen bietet Grasshopper externe Sicherheit durch Verschlüsselung der Agenten während der Migration. Mit Hilfe von Zertifikaten ist Grasshopper in der Lage, Agenten zu identifizieren und authentifizieren. Zum anderen erlauben interne Sicherheitsmechanismen Zugriffskontrolle innerhalb einer Agentur.

Registrierung Jede Agentur registriert alle local laufenden Agenten. Dadurch wird ein Beobachten und Kontrollieren der agenturinternen Prozesse ermöglicht. Ferner noch können sich so Agenten untereinander finden.

Persistenz Grasshopper kann die persistenten Daten innerhalb einer Agentur in gegebenen Zeitintervallen speichern. Im Falle eines Systemabsturzes kann der zuletzt gespeicherte Zustand wieder hergestellt werden.

Management Unter anderem ist der Managementservice für Erzeugen, Entfernen, Anhalten/Fortsetzen und Kopieren von Agenten verantwortlich. Über ein graphisches Interface können Administratoren immer in die Ausführung von Agenten eingreifen.

Transport Der Transportservice dient dazu, Agenten von der Agentur zu einer Zielagentur zu migrieren.

Kommunikation Dieser Service bietet einerseits die Basis für den Transport von Agenten. Andererseits muß er Interaktionen zwischen Agenten und nicht-Agenten-basierten Einheiten koordinieren.

Das Grundsystem von Grasshopper ist momentan für Windows und UNIX verfügbar. Eine auf WindowsCE/PocketPC zugeschnittene Version ist ebenfalls erhältlich.

Teil IV

Resümee

Kapitel 11

Zusammenfassung und Ausblick

11.1 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde die Mobile Agentenplattform SeMoA auf einem PDA, dem Compaq iPAQ 3660 realisiert. Dabei wurde versucht, möglichst den kompletten Funktionsumfang den SeMoA mit sich bringt zu realisieren.

Im ersten Teil wurde ein Überblick über die in dieser Arbeit berührten Themengebiete gegeben: Agententechnologie, hier speziell die Agentenplattform SeMoA und Vorraussetzungen, die von den untersuchten Virtual Machines erfüllt werden mußten. Es wurde weiterhin die entwickelte Testumgebung vorgestellt und erläutert, welche Funktionen damit geprüft worden sind.

Im zweitem Teil wurden die Bemühungen dargestellt, eine Agentenplattform auf einem Palm Vx zu realisieren. Dabei wurde zunächst auf das Gerät selber eingegangen und seine technischen Merkmale herausgestellt. Anschließend wurden einige der für dieses Gerät verfügbaren Virtual Machines beschrieben und untersucht und die jeweiligen Defizite der Virtual Machines aufgezeigt. Es wurde weiterhin begründet, warum SeMoA momentan nicht auf dem Palm Vx realisiert werden kann. Abschließend wurde eine Agentenplattform für den Palm Vx vorgestellt.

Im dritten Teil wurden, analog zu Teil 2, die Bemühungen dargstellt, SeMoA auf einen Compaq iPAQ 3660 zu portieren. Auch hier wurden erst die technischen Details und Möglichkeiten des Geräts dargstellt. Danach wurde auf den zwei bisher

erhältlichen Betriebssystemen die gängisten Virtual Machines beschrieben und untersucht. Dabei fand sich eine für die Realisierung der Aufgabenstellung dieser Diplomarbeit taugliche Virtual Machine unter dem Betriebssystem LinuxARM. Auch auf zwei Varianten dieses Betriebssystems wurde eingegangen. Zuletzt wurde kurz eine interessante Alternative zu den beiden bisher verfügbaren Betriebssystemen beschrieben. Am Ende des Teils wurden die portierte SeMoA-Plattform und eine selbstentwickelte Anwendung dargestellt, die Schwierigkeiten bei der Entwicklung erläutert und gefundene Lösungen beschrieben. Ferner wurden mögliche Erweiterungen der Anwendung aufgezeigt. Zuletzt wurden andere Agentenplattformen für den iPAQ vorgestellt.

11.2 Bewertung der gefundenen Lösung

SeMoA auf dem iPAQ bietet eine voll funktionsfähige Agentenplattform auf einem mobilen Gerät. Dabei wird die volle Sicherheitsarchitektur von SeMoA erhalten. Da SeMoA für leistungstarke Desktoprechner und Serversysteme entwickelt wurde, ist der Betrieb von SeMoA auf dem leistungsschwächerem iPAQ natürlich entsprechend langsamer. Aber im Gegensatz zu anderen Plattformen, die Abstriche bei Funktionalität und Sicherheit machen, um entsprechende Performance zu erzielen, kann SeMoA unverändert auf dem iPAQ ausgeführt werden und somit seine Stärken ausspielen. Das gewählte Linux-Betriebssystem als Basis stellt zudem noch einen flexiblen Arbeitsbereich zur Verfügung, der den individuellen Wünschen des Benutzers angepaßt werden kann. Desweiteren bietet die benutzte Virtual Machine den vollen Umfang des JDK 1.3.1 und somit können andere Java-Programme ebenfalls ohne Probleme benutzt werden. Dies alles sind Merkmale, die die im Rahmen dieser Arbeit gefundene Lösung sehr stark von vorhandenen Implementierungen abhebt.

Positiv anzumerken ist auch, daß sämtliche in der gefundenen Lösung benutzten Komponenten OpenSource-Projekte sind und somit Fehler schnell lokalisiert und korrigiert werden können. Ein Nachteil ist sicherlich der fehlende kommerzielle Support seitens der Hersteller des Betriebssystems und der Virtual Machine. Deshalb ist auch mehr Know-how in diesem Bereich für Betrieb und Installation notwendig. Es wurde jedoch versucht, dieses Know-how im Rahmen dieser Arbeit zu

bündeln und einem interessierten Fachpublikum zugänglich zu machen.

11.3 Ausblick

Die erarbeitete Lösung zeigt eine Möglichkeit, mit neuen Technologien, Standardgeräten maßgeschneiderte Aufgaben zuzuteilen. In Zukunft werden PDAs ähnlich weit verbreitet sein, wie Mobiltelefone derzeit. PDAs und Mobiltelefone werden zu einem Gerät verschmelzen, erste Ansätze zeigen die Smartphones [73]. Durch die ebenfalls ansteigende Leistungsfähigkeit mobiler Geräte werden wiederum neue Anwendungen realisierbar.

Ebenso wird sich das Konzept der Mobilen Agenten weiter verbreiten. Die Lösung zeigt die ersten Schritte zu einer *portablen* Mobilen Agentenplattform, für die in Zukunft weitere Anwendungsgebiete kommerziell erschlossen werden könnten.

Teil V

Anhang

Anhang A

Palm Vx

A.1 Installation: Sun Microsystems - JavaME

Die Installation von JavaME erfolgt einfach durch Aufspielen entsprechender `.PRC`-Dateien. Diese finden sich in dem von Sun Microsystems bereitgestelltem Archiv unter `j2me\_cldc/bin/kjava/palm`. Dort finden sich einige Beispiel- Spotlets, so werden die Anwendungen für die KVM auf dem Palm genannt, und eine Release- und eine Debug-Version der KVM. Das Archiv kann unter `http://www.sun.com/software/communitysource/j2me/cldc/` heruntergeladen werden.

Die Dateien `KVM.PRC`[1] und `KVMutil.PRC` müssen auf dem Palm oder in dem Emulator installiert werden.

Nun können die Beispiel-`.PRC`-Dateien oder eigene umgewandelte Klassen, auf dem selben Weg, installiert werden. Anschließend finden sie sich auf dem Desktop des Palm als Icon wieder und können, wie jedes andere Programm auch, gestartet werden. Mit Hilfe des `KVMutil.prc` können verschiedene Einstellungen an der Virtual Machine vorgenommen werden, wie zum Beispiel bereitgestellte Heapgröße. Auch das Speichern der Konsolenausgaben sowie deren Einsicht ist hier möglich. [64]

[1] Dies ist die Release-Version. Die Debug-Variante heißt `KVM_G.PRC`.

Um eigene Klassen in `.PRC`-Dateien zu wandeln, müssen folgende Schritte ausgeführt werden:

1. Kompilieren des Sourcecodes mit einem Java Compiler (`javac`)
2. Die kompilierten Klassen mit dem Tool `preverify` überprüfen. Das Tool ist Teil der *Java2 Micro Edition Reference Implementation.*
3. Mit Hilfe des `jar` Tools müssen alle benötigten Klassen in eine *JAR*-Datei gepackt werden.
4. Abschließend wird das erstellte *JAR* in eine `.PRC`-Datei gewandelt. Dazu gehört zu den Java2ME-Werkzeugen die Java-Applikation `makePalmApp` Beispielaufruf:

```
java -classpath ../bin/kjava/tools/palm/classes
palm.database.makePalmApp -v -version "1.1icon
icons/pong.bmp -bootclasspath ../bin/common/api/classes
-classpath Pong.jar;../bin/kjava/api/classes/pong.Pong
```

Die entstandende `.PRC` kann nun wie oben beschrieben auf den Palm transferiert und ausgeführt werden.

A.2 Installation: Kadasystems - KadaVM

A.2.1 Virtual Machine

Zunächst soll die Kada Virtual Machine auf dem Palm und eine entsprechende Entwicklungsumgebung auf dem Desktop-PC installiert werden. Hierfür wird benötigt:

- Palm Desktop (aktuelle Version erhätlich bei `www.palm.com`)
- Java SDK 1.1.8 (besser Java2 JDK 1.3 oder neuer), die KadaTools benötigen bereits eine Java2 kompatible Virtual Machine. Kada Systems bereitet auch eine Java2 kompatible Version der KadaVM vor.
- Palm Emulator (zu Debugzwecken)

- ein ZIP-kompatibles Kompressionsprogramm, beispielsweise WinZip
- Microsoft Windows NT 4 Service Pack 4 oder neuer wird empfohlen
- eine Version der KadaVM

Die Evaluationsversion der Virtual Machine von Kada Systems liegt unter `http://www.kadasystems.com`bereit. Die aktuelleren Versionen der KadaVM werden als ausführbare Archive ausgeliefert und entpacken sich automatisch bei deren Ausführung. Ein entsprechendes Verzeichnis kann gewählt werden. Desweiteren sollte das Java Development Kit (JDK) installiert werden. Einige Umgebungsvariablen müssen vor dem Betrieb von Kada angepaßt werden. So sollte die `PATH`-Variable sowohl das Kada-Verzeichnis, als auch das des Palm Desktop enthalten. Dann muss eine Variable `KADA_CLASSPATH` und `KADA_HOME` angelegt werden.

Beispiel:

```
set PATH=%PATH%;D:\KADA\BIN\WIN32;C:\PALM
set KADA_HOME=D:\KADA
set KADA_CLASSPATH=.;D:\KADA\LIB\KADAAPI.ZIP;D:\KADA\KADATO
OLS\KADATOOLS.ZIP
```

Als nächstes muss der *KadaInstaller* konfiguriert werden. Dazu müssen folgende Dateien aus dem `kada/kadatools`-Verzeichnis in das `C:/PALM`-Verzeichnis kopiert werden. Es wird empfohlen, stets die neuere Dateiversion zu verwenden, sollten eine oder mehrere Dateien bereits existieren:

- Condcfg.exe
- CondMgr.dll
- HSAPI.dll
- KadaInstaller.dll

Nun sollte ein ggf. aktiver HotSync-Manager geschlossen werden und das Programm *Condcfg.exe* gestartet werden. Ein der Abbildung A.1 ähnliches Fenster

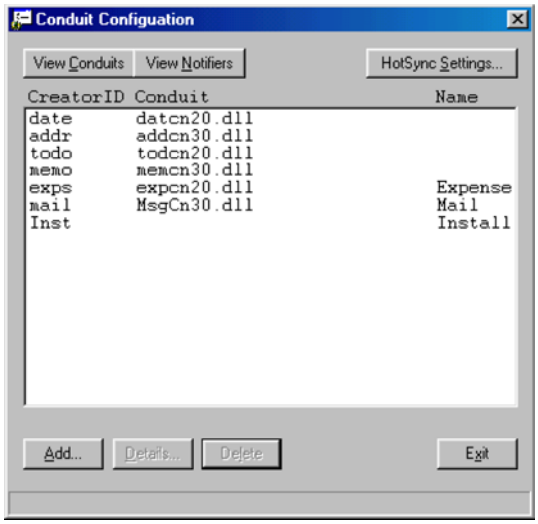

Abbildung A.1: Conduit Configuration

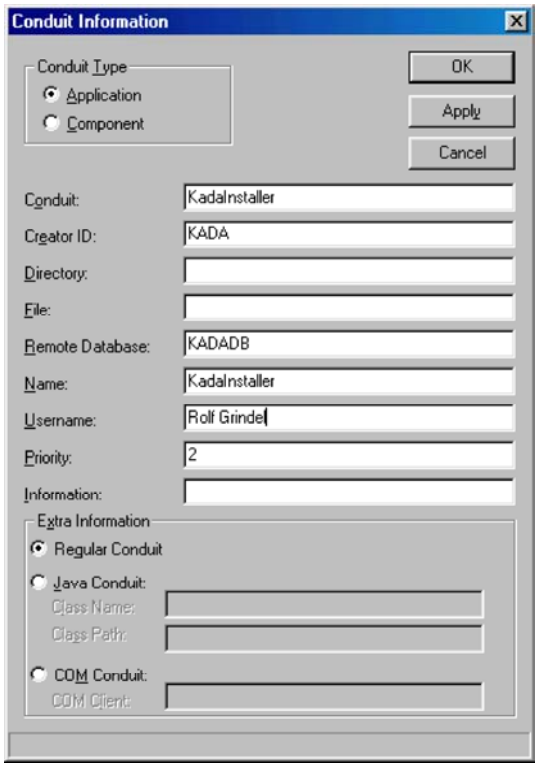

Abbildung A.2: Konfiguration des KadaInstaller-Conduit

erscheint. Dort wird dann über „Add“ ein entsprechender Conduit angelegt. Abbildung A.2 zeigt die dort nötigen Einstellungen, wobei der Benutzername auf den Palm Desktop abgestimmt sein sollte.

Nun folgt die Installation der Virtual Machine auf dem Palm selbst, oder im Emulator. Hierzu wird über den Palm Desktop über *Programm hinzufügen* die entsprechende *.PRC*-Datei installiert. Die entsprechende *KadaVM.PRC*-Datei findet sich im `kada/bin/palm`-Verzeichnis. Dort liegt entweder die Debugversion (im *Debug*-Verzeichnis) oder die Release (im *Release*-Verzeichnis) bereit. Nach Wahl der entsprechende Virtual Machine wird eine HotSync-Session gestartet und somit die KadaVm auf dem Palm oder Emulator[2] installiert. Anschliessend sollte der Palm via Reset neu gestartet werden. Dies schließt die Installation der KadaVM ab. [28]

A.2.2 Installation der Testumgebung unter KadaVM

Bei der KadaVM werden alle benötigten Klassen in einem ZIP-Archiv zusammengestellt und dieses dann via HotSync auf den Palm transferiert, wo die entsprechende Installationsroutine die Klassen dort in einer Datenbank ablegt. Eine erneute Installation überschreibt grundsätzlich alle in dieser Datenbank vorhandenen Klassen, weswegen man stets alle Klassen, die benötigt werden in diesem ZIP-Archiv halten sollte.

Die KadaVM bietet desweiteren einen Mechanismus an, der vorhandene Klassenbibliotheken nach nicht benötigten Klassen und innerhalb der Klassen nach Methoden sucht und diese dann aus dem Archiv entfernt, so daß Speicherplatz gespart werden kann. Es wird vom Autor aber nicht empfohlen dieses Tool einzusetzen, da es vorkommen kann, daß Klassen oder Methoden entfernt werden, die sehr wohl benötigt werden.

In dem ZIP-Archiv müssen folgende Komponenten enthalten sein:

- KadaVM-API-Klassen, werden von der Kada bereitgestellt und müssen in das Archiv gepackt werden

[2] Beim Emulator besteht auch die Möglichkeit die .PRC-Dateien via Drag 'n' Drop zu installieren.

- eigene Klassen
- alle sonst benötigten Klassen, die durch eigene Klassen benutzt werden

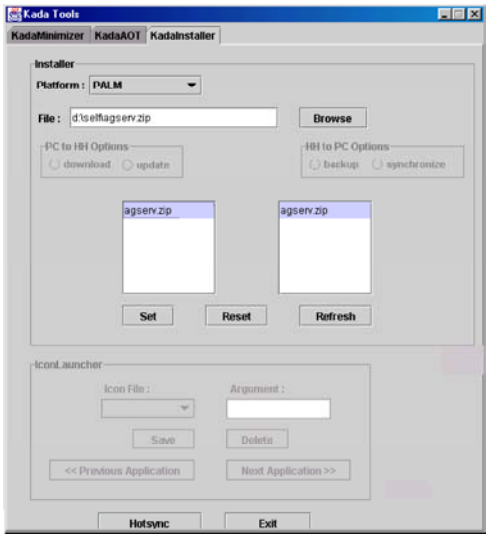

Abbildung A.3: KadaInstaller

Nachdem man das ZIP-Archiv erstellt hat, startet man das zu den *KadaTools* gehörende Programm *KadaInstaller* (siehe Abbildung A.3). Hier wählt man dann das zu übertragende Klassenarchiv und bestätigt mit „Set“ und beendet den Installer mit „Exit“.

Nun startet man einen HotSync-Transfer und die Klassen werden in die Datenbank übertragen. Anschliessend können diese von der KadaVM benutzt werden. [27]

Anhang B

Compaq iPAQ

B.1 Installation: Insignia Solutions Jeode

Die Installation von Jeode Runtime erweist sich als äußerst einfach. Jeode wird als auf Win32-Systemen ausführbare Datei ausgeliefert. Es wird eine vorhandene ActiveSync-Verbindung zwischen dem Desktop-PC und dem iPAQ vorrausgesetzt. Die Installationsdatei übernimmt nach ihrem Start die Arbeit und kopiert über die ActiveSync-Verbindung die nötigen Dateien auf den iPAQ.

Jeode installiert die Dateien in das `/Windows`-Verzeichnis und in zugehörige Unterverzeichnisse. Die Deinstallation erfolgt, über die Windows-eigenen Deinstallationsmechanismen unter Start → Einstellungen → System. Dort *Insignia Solutions Jeode* und *JeodeForArmPocketPCInstaller* entfernen. Sollte die Deinstallation fehlschlagen, so kann es sein, daß noch zu entfernende Dateien benutzt werden. In diesem Fall einfach den iPAQ resetten und erneut probieren. Die Installation beinhaltet sowohl die Plugin-Komponente für den Internet Explorer und eine Standalone-Version. Der Internet Explorer startet die Virtual Machine bei Bedarf selbsttätig. Um die Virtual Machine manuell zu starten, muß das Programm `/Windows/evm.exe` gestartet werden. Um direkt Klassen auszuführen, empfiehlt Insignia Solutions, eine Verknüpfung anzulegen und dort die eigenen Klassen als Startparameter mit anzugeben. [61]

B.2 Installation: LISA mLinux

Um mLinux auf dem iPAQ zu installieren, muß zunächst das ausgelieferte Betriebssystem WindowsCE gelöscht und ein neuer Bootloader — die Einheit, die nach einem Reset gestartet wird und die das Booten (Laden) des Betriebssystems übernimmt — überschrieben werden.

WARNUNG: Sollte beim Schreiben des Bootloadersdes in das FlashROM des iPAQs was schieflaufen, so wird der iPAQ nutzlos und kann nur vom Compaq-Servicecenter reaktiviert werden.

B.2.1 Vorraussetzungen

Für einen WindowsCE basierten iPAQ wird benötigt:

- ein Windows 95/98/98SE/ME/NT/2000/XP Rechner, auf dem die ActiveSync Software läuft, die mit dem iPAQ ausgeliefert wird.
- eine serielle Verbindung zum iPAQ. USB wird derzeit nicht unterstützt. Sollte die Basisstation mit der seriellen Verbindung nicht verfügbar sein, so beschreibt MONTA im Onlineforum von Handhelds.org [47] eine Anleitung, wie ein solches Kabel selbst gebastelt werden kann, allerdings unter Verlust der Gewährleistung für die USB-Basisstation.
- ein Arbeitsverzeichnis: Hier sollten alle nötigen Dateien gesammelt werden. Diese können entweder auf der LISA mLinux CD gefunden oder vom LISAs FTP-Server [39] heruntergeladen werden. Es empfiehlt sich, die Integrität der Dateien mittels der MD5-Prüfsummen zu überprüfen.

  Folgende Dateien sollten sich dort befinden:

  - `OSloader-1_3_0.exe`
  - `OSloader-1_3_0.exe.md5sum`
  - `bootldr-c002-2_9_5`
  - `bootldr-c002-2_9_5.md5sum`
  - `bootldr-000-03062001`

  - `bootldr-000-03062001.md5sum`
  - `zImage-2.4.2-rmk1-np2`
  - `zImage-2.4.2-rmk1-np2.md5sum`
  - `init-mLinux-0.9.cramfs`
  - `init-mLinux-0.9.cramfs.md5sum`
  - `root-mLinux-0.9.cramfs`
  - `root-mLinux-0.9.cramfs.md5sum`
  - `usr-mLinux-0.9.cramfs`
  - `usr-mLinux-0.9.cramfs.md5sum`

- ein Terminalprogramm (Hyperterm, Minicom, etc.).

B.2.2 Sicherheitskopie von WindowsCE

Nachdem sichergestellt ist, daß alles Nötige vorhanden ist, sollte als nächstes eine Sicherheitskopie von WindowsCE angelegt werden, um bei Bedarf das Original Betriebssystem zur Verfügung zu haben.

1. Den iPAQ mit dem Windows Rechner über den seriellen Anschluß verbinden
2. Den iPAQ so konfigurieren, daß er für ActiveSync die serielle Verbindung benutzt
   - Im Start-Menü (das Symbol mit der Microsoft Flagge) *Einstellungen* auswählen
   - Verbindungen und dort das PC Symbol wählen
   - *Automatisch synchronisieren, wenn serielle Verbindung entdeckt wurde* aktivieren.
   - Von *USB* auf *115200 Standard* stellen
   - Mit *OK* (rechts oben in der Ecke) bestätigen
3. Die ActiveSync Applikation starten, um den PC mit dem iPAQ zu verbinden.

4. Die `OSloader-1.3.0.exe` vom PC auf den iPAQ kopieren. Die Meldung, daß evtl. Konvertierungen nötig sind, kann ignoriert werden.

5. Die Datei `bootldr-c002-2.9.5` in `bootldr` umbenennen und dann in das Hauptverzeichnis vom iPAQ kopieren (`Mein PocketPC`).

6. Die Anwendung `OSloader-1.3.0` auf dem iPAQ ausführen.

7. *Tools* → *Flash* → *Save to files* auswählen. Es werden nun nacheinander vier Dateien mit jeweils 4 MB Größe angelegt. Nach jeder angelegten Datei, diese auf den Desktop Computer kopieren und anschließend vom iPAQ löschen. Während die Dateien auf dem iPAQ angelegt werden, scheint es, als wäre er eingefroren und reagiert auf keinerlei Eingabe.

Bis hierhin ist noch nichts passiert, was die Funktionsfähigkeit des iPAQ einschränken könnte. Ein Reset führt weiterhin dazu, daß der Bootloader WindowsCE startet. Deshalb wird nun als nächstes genau dieser ersetzt.

B.2.3 Bootloader

Es wird davon ausgegangen, daß bisher kein anderer Bootloader installiert war und demnach noch WindowsCE gebootet wird. Sollte dies nicht mehr der Fall sein, weil beispielsweise bereits eine Linux-Installation erfolgte, so fahren Sie bitte an der Stelle fort, an der das Terminalprogramm zum Einsatz kommt. Es sei nochmal darauf hingewiesen, daß das Austauschen des Bootloaders mit dem Risiko verbunden ist, den iPAQ unbrauchbar zu machen — so fern beim Schreiben des Bootloaders etwas schief gehen sollte.

1. `OSloader-1.3.0.exe` auf dem iPAQ starten und *Tools* → *Flash* → *Run after loading from file* auswählen. An dieser Stelle sollte der iPAQ-Bildschirm schwarz werden.

2. Die Verbindung in der ActiveSync Applikation auf dem Desktop Computer trennen, da sie die serielle Schnittstelle belegt, die gleich benötigt wird. Sollte für die nächsten Installationsschritte weiterhin Windows zum Einsatz kommen (z.B. wenn Sie Hyperterm als Terminalprogamm benutzen), sollten folgende Schritte durchgeführt werden:

- Rechtsklick auf das ActiveSync-Logo in der Statuszeile.
- *Verbindungseinstellungen* wählen.
- *Serielle oder Infrarot Verbindung erlauben* für die benutzte serielle Schnittstelle (COM-Port) ausschalten.

3. Nun das Terminalprogramm starten. Folgende Einstellungen für die Verbindung einstellen:
   - *115200 8N1* (115200 Baud, 8 Bit, keine Parität, 1 Stopbit)
   - Flußkontrolle: *keine*
4. Drücken der Eingabetaste in dem Terminalprogramm sollte einen `boot>`-Prompt erscheinen lassen. Mit `help` können Sie ein Liste der möglichen Kommandos erhalten.
5. Nun zu dem gefährlichem Teil: Nach dem `boot>`-Prompt *load bootldr* eingeben und die Datei `bootldr-0000-03062001` via XModem-Upload hochladen. XModem ist ein in der Datenfernübertragung gängiges Flußprotokoll und sollt von jedem Terminalprogramm beherrscht werden. Auftretende Übertragungsfehler können erkannt und durch wiederholtes Übertragen korrigiert werden. [49]
6. Als Rückmeldung sollte `verifying... done.` erscheinen. Das Ladeprogramm hat eine simple Überprüfung eingebaut, die sicherstellt, daß nur Bootloader an die Bootloader-Speicherstelle im Flashspeicher geschrieben werden.
7. Sicherstellen, daß der Bootloader Sektor geschützt ist
   - `qflash 2` eingeben.
   - Als Ausgabe sollte `00010001` erscheinen.
   - Sollte die Ausgabe nicht erscheinen, mit `pflash 0 0xffff 1` den Sektor schützen.
8. Nun muß der iPAQ neu gebootet werden. Dazu entweder den *Reset*-Knopf auf der Unterseite drücken oder an dem `boot>`-Prompt `reset` eingeben.

Nun sollte das Bootloader-Logo auf dem iPAQ erscheinen und einige Meldungen in dem Terminalfenster. Sollte soweit alles gut gegangen sein, dann ist der gefährlichste Schritt überstanden. Über die Meldung *Corrupt kernel image* brauchen Sie sich momentan noch keine Gedanken machen, da ja noch kein Kernel installiert wurde, was aber im nächsten Schritt nachgeholt wird.

B.2.4 Kernel und Filesystem

Nachdem der Bootloader installiert und gebootet wurde, folgt der nächste Schritt, die Installation des Kernels.

1. Das Laden des Kernels geht ähnlich dem Laden des Bootloaders. An dem `boot>`-Prompt `load kernel` eingeben und die Datei `zImage2.4.2-rmk1-np2` via XModem Transfer hochladen.
2. Wenn das Hochladen beendet ist, sollten `Erasing, Writing, Verifying flash` Meldungen erscheinen.
3. Als nächstes werden die verschiedenen CRAMFS[1]-Images in verschiedene Flashbereiche geladen. Folgende Befehle sollten wie üblich am 'boot>-Prompt erfolgen und das Hochladen jeweils via XModem-Transfer:
   - `load flash 0x100000` die Datei `init-mLinux-0.9.cramfs` hochladen. Dies wird ungefähr vier Minuten dauern. Nach dem Upload erscheinen wieder `Erasing, Writing, Verifying flash` Meldungen.
   - `load flash 0x200000` die Datei `root-mLinux-0.9.cramfs` hochladen. Dies wird etwa zehn Minuten beanspruchen. Auch hiernach erscheinen wieder `Erasing, Writing, Verifying flash` Meldungen.
   - `load flash 0x500000` die Datei `usr-mLinux-0.9.cramfs` hochladen. Die Dauer des Transfers beträgt ca. 20 Minuten. Abschließend werden wieder `Erasing, Writing, Verifying flash` Meldungen ausgegeben.

[1] Compressed RAM FileSystem

4. Nun werden noch ein paar Einstellungen gesetzt. Folgende Befehle, jeweils am `boot>`-Prompt eingeben:

   - `set linuxargs 'noinitrd root=/dev/mtdblock4 init=/linuxrc console=ttySA0 mem=64M'`
   - `set copy_ramdisk 0x0`
   - `set baudrate 115200`
   - `params save`
   - `boot`

Der letzte Befehl weist den iPAQ an, den Linux Kernel zu booten. In der Installationsanleitung von LISA mLinux (vgl. [40]) gibt es noch einen skriptgesteuerten Weg, der hier nicht näher dargelegt werden soll. Diese ausführliche Beschreibung wurde gewählt, da sie im Prinzip für aller verschiedenen Linuxderivate für den iPAQ gilt.

Anhang C

Handhelds.org Familiar

C.1 Installation

C.1.1 Vorraussetzungen

Für einen WindowsCE basierten iPAQ wird benötigt:

- ein Windows 95/98/98SE/ME/NT/2000/XP Rechner, auf dem die ActiveSync Software läuft, die mit dem iPAQ ausgeliefert wird.
- eine serielle Verbindung zum iPAQ. USB wird derzeit nicht unterstützt. Sollte die Basisstation mit der seriellen Verbindung nicht verfügbar sein, so beschreibt MONTA im Onlineforum von Handhelds.org [47] eine Anleitung, wie ein solches Kabel selbst gebastelt werden kann, allerdings unter Verlust der Gewährleistung für die USB-Basisstation.
- ein Arbeitsverzeichnis: Hier sollten alle nötigen Dateien gesammelt werden. Folgende Dateien sollten sich dort befinden:
  - nur bei WindowsCE System: `OSloader-1_3_0.exe` (Programm, daß einen Bootloader laden kann)
  - nur bei WindowsCE System: `bootldr-c002-2_9_5` (ein Bootloader für OSloader)

  - ein Bootloader, Beispiel: `bootldr-2.15.15` (wird in die bootldr-Sektion geflasht)
  - das Root- und Kernel-Image, hier: `ipaq-familiar_0.5-semoa.jffs2` (dies ist ein selbst erstelltes Image, in dem einige Konfiguration und Installation von beispielsweise SeMoA bereits vorgenommen wurde. Eine Java VM ist auf Grund ihrer Größe nicht eingebunden.)

- ein Terminalprogramm (Hyperterm, Minicom et cetera).

C.1.2 Sicherheitskopie von WindowsCE

Die Sicherheitskopie von WindowsCE läuft identisch ab, wie in B.2.2. Zur Vollständigkeit und auf Grund der Übersicht, werden hier die einzelnen Punkte kurz wiederholt.

1. den iPAQ mit dem Windows Rechner über den seriellen Anschluß verbinden
2. ActiveSync für die Benutzung der seriellen Schnittstelle konfigurieren
   - *Einstellungen* im Start-Menü (das Symbol mit der Microsoft Flagge) auswählen
   - Verbindungen wählen und dort das PC Symbol
   - *Automatisch synchronisieren, wenn serielle Verbindung entdeckt wurde* aktivieren.
   - Von *USB* auf *115200 Standard* stellen
   - Mit *OK* (rechts oben in der Ecke) bestätigen
3. ActiveSync Applikation starten
4. `OSloader-1.3.0.exe` vom PC auf den iPAQ kopieren
5. die Datei `bootldr-c002-2.9.5` in `bootldr` umbenennen und in das Hauptverzeichnis vom iPAQ kopieren
6. die Datei `OSloader-1.3.0` ausführen
7. *Tools→Flash→Save to files* wählen und die vier Dateien auf den PC kopieren

C.1.3 Bootloader

Auch dies unterscheidet sich nicht sehr von der bereits beschriebenen Methode. Ausnahmen bilden jedoch die verwendeten Versionen der verwendeten Komponenten. Desweiteren wird davon ausgegangen, daß das Terminalprogramm richtig konfiguriert ist und ein `boot>`-Prompt erscheint.

Die hier verwendete Komposition von Betriebssystem und Applikationen basiert auf Familiar v0.5. Diese benutzt jedoch Eigenschaften, die einen neueren Bootloader benötigen, als das bei LISA mLinux 0.9 der Fall ist. Es wird der Bootloader `OHH/CRL bootldr v2.15.15` oder eine neuere Version empfohlen. Sollte sich dieser bereits auf dem iPAQ befinden, so kann dieser Teil übersprungen werden. Vor dem Start sollte sichergestellt werden, daß der iPAQ gut geladen und an eine Konstantstromquelle angeschlossen ist. *Warnung:* Unter keinen Umständen den iPAQ während des Prozesses neu starten. Es besteht die Möglichkeit, daß sich der iPAQ in einen instabilen Zustand begibt und demnach unbrauchbar wird.

- Am `boot>`-Prompt `load bootldr` eingeben.
- Den Bootloader `bootldr-2.15.15` via XModem-Transfer hochladen.

Beispiel:
```
boot> load bootldr loading flash region bootldr
using xmodem
ready for xmodem download..
BSD sum value is: 00000000
programming flash...
unlocking boot sector of flash
Protect=00000000
erasing ...
Erasing sector 00000000
writing flash..
addr: 00000000 data: EA00008E
addr: 00010000 data: E1A0C00D
verifying ... done.
smartAddress :00000000
```

```
limitAddress :00018980
Protecting sector 00000000
Protect=00010001
```

Hinweis: Das Beispiel ist aus der Originalinstallationsanleitung übernommen [15]. Es ist nicht schlimm, wenn sich die Zahlen unterscheiden. Wichtig ist, daß die `'erasing .. writing flash .. verifying .. done'` Schritte erfolgreich sind. Dies gilt im übrigen auch für die folgenden Beispiele.

Nun sollte noch überprüft werden, ob der Sektor des Bootloaders geschützt wurde.

- Am `boot>`-Prompt `qflash 2` eingeben.

Sollte das Ergebnis `00010001` sein, so ist alles in Ordnung. Andernfalls den Sektor manuell schützen, in dem `pflash 0 0xffff 1` eingeben wird. Es richtet auch keinen Schaden an, wenn dieser Schritt bei bereits geschütztem Sektor durchgeführt wird.

An dieser Stelle muß der iPAQ neu gestartet werden. Dies geschieht mit Hilfe des Stifts, in dem der Reset-Knopf an der Unterseite gedrückt wird. Anschließend den iPAQ wieder auf die Basisstation stellen und wenn das Bootloader-Logo erscheint, die Leertaste drücken. So müßte wieder der `boot>`-Prompt erscheinen.

Als nächstes sollten einige Parameter gesetzt werden, wie es auch schon bei LISA mLinux 0.9 der Fall war. Allerdings unterscheiden sich die Parameter einwenig. Geben Sie folgende Befehle jeweils am `boot>`-Prompt ein.

- `params reset` — Um gegebenenfalls bereits eingegebene Parameter zu löschen bzw. mit den Standardwerten zu überschreiben.
- `partitions reset` — Hier werden die verschiedenen Partitionsdefinitionen durch Standardwerte ersetzt.
- `set linuxargs 'noinitrd root=/dev/mtdblock/2 init=/linuxrc console=ttySA0'`
- `params save` — Einstellungen sichern

Beispiel: `boot>` **`params reset`**

```
setting params to default values
boot> partition reset
argv[1]=reset
defining partition: bootldr
defining partition: params
defining partition: root
boot> set linuxargs \noinitrd root=/dev/mtdblock/2 init=/li
nuxrc console=ttySSA0\
setting param=linuxargs to value=noinitrd root=/dev/mtdblock
/2 init=/linux rc console=ttySSA0
boot> params save
bootldr: set linuxargs \noinitrd root=/dev/mtdblock/2 init=
/linuxrc console=ttySSA0\

programming flash...erasing ...
Erasing sector 00040000
writing flash..
addr: 00040000 data: 646C7470
verifying ... done.
boot>
```

Dies schließt die Installation des Bootloaders ab. Als nächstes werden die Dateisysteme geschrieben.

C.1.4 Kernel und Root-Filesystem

Mit der Einführung von Familiar v0.5 *Pre-Release* gibt es keine Unterscheidung mehr zwischen Kernel- und Root-Dateisystem. Beide liegen innerhalb des gleichen Blocks und so können auch beide mit Hilfe eines einzigen FlashROM-Abbildes geschrieben werden. Im Rahmen dieser Arbeit wurde ein benutzerdefiniertes FlashROM-Abbild erzeugt. Dieses wird in diesem Schritt nun auf den iPAQ geladen.

- Am `boot>`-Prompt `load root` eingeben.

- Via XModem-Transfer und die Datei `ipaq-familiar_0.5-semoa.jffs2` hochladen. Das Hochladen benötigt etwa 45 Minuten (vergleiche hier die Gesamtdauer bei der LISA mLinux Installation; es werden etwa gleiche Datenmengen übertragen, hier jedoch auf einmal).

Es sollten dem Beispiel ähnliche Ausgaben erscheinen:

```
boot> load root
loading flash region root
using xmodem
ready for xmodem download..
Erasing sector 00140000
Erasing sector 00180000
Erasing sector 001C0000
Erasing sector 00200000
.
.
.
writing flash..
addr: 00100000 data: E0021985
addr: 00110000 data: E3BAD617
addr: 00120000 data: 0FA1F57B
addr: 00130000 data: 9343AEEB
.
.
.
verifying ... formating ... done.
boot>
```

Nun sollte das gerade geschriebene System gestartet werden. Dies wird durch den `boot` Befehl erreicht.

- Am `boot>`-Prompt: `boot` eingeben.

Linux sollte nun verschiedene Meldungen ausgeben und Daemons starten. Wenn soweit alles gut gegangen ist, sollte ein '`login:` '-Prompt erscheinen. Ein paar Einstellungen sollten noch vorgenommen werden. So muß zum Beispiel noch das Datum und die Uhrzeit eingestellt werden. Dies kann entweder manuell[1] oder automatisch über das Netzwerk[2] erfolgen. Um dies erledigen zu können, melden Sie sich als Benutzer `root` an. Das Passwort lautet `rootme`.

C.2 Aironet 4500 Wireless LAN-Karte

Die Aironet 4500 Wireless LAN-Karte wird dank neuem Kernel der Familiar v0.5 selbsttätig erkannt und entsprechende Module bei ihrem Einschieben oder Vorhandensein automatisch geladen. Was noch konfiguriert werden muß ist die ESSID[3]. Die ESSID identifiziert das Wireless-LAN. Sie muß mit der eingetragenen ESSID am Zugangspunkt übereinstimmen. Zugelassen sind bis zu 32 Zeichen, deren Groß-/Kleinschreibung unterschieden wird. [60]. Die Fraunhofer Gesellschaft in Darmstadt benutzt momentan als ESSID `tsunami`. Nach dem Laden der Treibermodule wird die ESSID mit `iwconfig eth0 essid tsunami` gesetzt. Es ist darauf zu achten, daß die Karte beim Laden der Treiber möglichst eine Basisstation finden kann.

Durch folgende Befehle kann läßt sich der ordnungsmäßige Betrieb der Karte feststellen:

1. `iwconfig eth0`, oder
2. `cat /proc/driver/aironet/eth0/Status`

Eine ausführliche Installationsanleitung für Wireless-LAN unter Linux findet sich auf der Homepage von MARKUS HOLTMANN [19].

[1] über den Befehl `date`

[2] Beispielsweise mit `ntpdate clepsydra.research.compaq.com`, eine funktionierende Netzwerkverbindung vorrausgesetzt

[3] Extended Service Set Identifier

C.3 Blackdowns Java Runtime 1.3.1RC1 für Linux

Die Installation der Blackdown Java Runtime 1.3.1RC1 ist ähnlich einfach, wie die Installation der Jeode Runtime unter WindowsCE. Die Virtual Machine wird in zwei Teilen geliefert: *der Virtual Machine* und *fehlenden Libraries*, die nicht im eigentlichen Sinne zur Virtual Machine gehören, jedoch von ihr benötigt werden. Beides liegt auf Blackdowns FTP-Server [3] zum Download bereit:

- `j2re-1.3.1-RC1-linux-arm.tar.gz` und
- `additional-ipaq-stuff.tar.gz`

Beide Dateien lassen sich mit `tar xvfz dateiname` auspacken. Das Archiv `additional-ipaq-stuff.tar.gz` enthält ein `lib` und ein `usr` Verzeichnis, in dem verschiedene Libraries liegen. Idealerweise sollten diese Dateien in die entsprechenden Unterverzeichnisse des iPAQ-Linux kopiert oder verschoben werden (*Hinweis:* symbolische Links sollten aus Speicherplatzgründen erhalten bleiben). Notfalls können die im `lib`-Verzeichnis liegenden Dateien aber auch in das `lib/armv4l/`-Verzeichnis des Virtual Machine Archivs gelegt werden. Die Dateien und Unterverzeichnisse des `usr/X11R6/lib`-Verzeichnis ebenso.

Die Dateien lassen sich beispielsweise via ZModem und der seriellen Schnittstelle hochladen oder wesentlich komfortabler über Netzwerk und `scp`. Auch die Einrichtung eines FTP-Servers auf dem iPAQ ist möglich.

Beispiel:

```
ipaq:~ # cd /tmp
ipaq:/tmp # scp rolf@giedi:~/ipaq/j2re-1.3.1-RC1-linux-arm.
tar.gz .
   ...
ipaq:/tmp # tar xvfz j2re-1.3.1-RC1-linux-arm.tar.gz
ipaq:/tmp # rm j2re-1.3.1-RC1-linux-arm.tar.gz
ipaq:/tmp # scp rolf@giedi:~/ipaq/additional-ipaq-stuff.tar
.gz .
   ...
ipaq:/tmp # tar xvfz additional-ipaq-stuff.tar.gz
```

```
ipaq:/tmp # rm additional-ipaq-stuff.tar.gz
ipaq:/tmp # cd j2re1.3.1/lib
ipaq:/tmp/j2re1.3.1/lib # mv /tmp/lib/* .
ipaq:/tmp/j2re1.3.1/lib # cd armv4l/lib
ipaq:/tmp/j2re1.3.1/lib/armv4l/lib # mv /tmp/usr/X11R6/lib/
* .
ipaq:/tmp/j2re1.3.1/lib/armv4l/lib # cd /tmp
ipaq:/tmp/ # rm -r lib
ipaq:/tmp/ # rm -r usr
ipaq:/tmp/ # mv j2re1.3.1 /var/
ipaq:/tmp/ # ln -s /var/j2re1.3.1/bin/java /bin/java
```

C.4 Installation der Testumgebung

Die Installation des Testprogramms *AgServ* sollte keinerlei Probleme bereiten. Es müssen einfach die nötigen *.class*-Dateien auf den iPAQ kopiert werden. Die Kompilierung von Java-Sourcecode ist auf dem iPAQ derzeit nicht möglich, da kein Java2 SDK zur Verfügung steht.

Die Dateien *Server.class, Listen.class* und *Agent.class (und Agent$1.class im GUI-Fall)* müssen auf den iPAQ kopiert werden.

Beispiel:

```
ipaq:~ # mkdir AgServ
ipaq:~ # cd AgServ
ipaq:~/AgServ # scp rolf@giedi:~/Server.class .
   ...
ipaq:~/AgServ # scp rolf@giedi:~/Listen.class .
   ...
ipaq:~/AgServ # scp rolf@giedi:~/Agent*.class .
```

C.5 SeMoA

SeMoA läßt sich entweder von der Webseite[4] herunterladen oder einfach kopieren. SeMoA ist in dem selbsterstelltem Flashabbild bereits enthalten.

[4] www.semoa.org

Beispiel:

```
ipaq:~ # mkdir semoa
ipaq:~ # cd semoa
ipaq:~/semoa # scp rolf@giedi:~/semoa/semoa.tar.gz .
   ...
ipaq:~/semoa # tar xvfz semoa.tar.gz
   ...
ipaq:~/semoa # rm semoa.tar.gz
ipaq:~/semoa # bin/semoa
```

Der letzte Aufruf startet SeMoA, ein installiertes Java vorrausgesetzt.

C.6 Erzeugen eigener JFFS2-Flashabbilder

Flashabbilder sind Archive, die mit Hilfe des *Bootloaders* in den Flashspeicher des iPAQ geschrieben werden können. Sie enthalten alle benötigten Dateien und Daten, die für den Betrieb nötig sind.

Das Erzeugen eigener JFFS2[5]-Flashabbilder ist denkbar einfach. Man benötigt dazu am Besten das Grundpaket der zu installierenden Familiar Version (vgl. [16]), seine eigenen Dateien, sowie evtl veränderte Konfigurationsdateien. Desweiteren wird das Tool *mkfs.jffs2* benötigt (vgl. [46]).

Folgende Schritte sind erforderlich, um ein eigenes JFFS2-Abbild zu erzeugen:

- Alle Dateien und Verzeichnisse, die in dem Abbild vorkommen sollen in ein Verzeichnis kopieren.
- Den Besitzer der Dateien auf *root* ändern: *find ./ | xargs chown root.root*
- mkfs.jffs2 -o Ausgabedatei.jffs2 -d Verzeichnis -p -e 0x40000 ausführen. Beispiel: *mkfs.jffs2 -o new-ipaq-image.jffs2 -d task-bootstrap -p -e 0x40000*

Hinweis: Statt der 0x40000 ist 0x20000 bei monochromen iPAQs anzugeben.

[5] Journalised Flash Filesystem 2

C.7 Inhalt des ipaq-familiar_0.5-semoa.jffs2-Flashimages

In dem selbsterstelltem JFFS2-Image *ipaq-familiar_0.5-semoa.jffs2* sind alle notwendigen Pakete des Familiar-Linux installiert und deren Konfiguration, wie beispielsweise die Einrichtung des SSH-Daemons, bereits erfolgt. Des weiteren wurde SeMoA und AgServ in ein Unterverzeichnis des Hauptverzeichnis des Superusers gelegt. Nach der Installation von Java und Setzen eines entsprechenden symbolischen Verweises kann SeMoA direkt benutzt werden.

Die Konfiguration ist in soweit vollständig, so daß die Aironet Wireless LAN-Karte auch direkt nach Einschieben und setzen der ESSID benutzt werden kann.

Folgende Dateien wurden im einzelnen verändert oder angepaßt:

- `/bin/java`: Symbolischer Link nach /var/j2re1.3.1/java
- `/etc/hostname`: Name für den Rechner (hier: ipaq)
- `/etc/hosts`: Einträge für Maschinen im LAN (giedi, arrakis, ipaq, laptop, etc.)
- `/etc/ipkg.conf`: Quellen für IPKG-Pakete (momentan lokaler Webserver; original Adresse im Kommentar enthalten)
- `/etc/modules.conf`: Anpassung der PPP-Modulaliasnamen
- `/etc/passwd`: Eintragung eines Benutzers für PPP via Windows 9X.
- `/etc/ppp/options`: Konfiguration des PPP-Zugangs
- `/etc/ssh/`: Generierung des nötigen Schlüssel.
- `/etc/X11/blackbox/blackbox-menu`: Menü des Fenstermanagers
- `/home/ppp95/.ppprc`: Konfiguration für PPP via Windows 9X
- `/root/`: Installation von AgServ und SeMoA

Diese Schritte sind bereits in dem selbsterstlltem Flashabbild vollzogen worden. Sie sind nur notwendig, wenn ein neues, den Funktionsumfang entsprechendes Abbild erzeugt werden soll.

Tabelle C.1 gibt derzeit verfügbaren Pakete und eine kurze Beschreibung an. Ihr kann ferner entnommen werden, ob diese Pakete in der selbstzusammengestellten Installation zum Einsatz kommen. Weitere Pakete finden sich bei Blackdown im sogenannten „unstable-feed“. Diese Pakete befinden sich noch in der Entwicklung und liegen dort zum Test bereit.

Paketname	Beschreibung	installiert
3c574-modules-2.4.7-rmk3-np1-devfs	3c574 modules for kernel 2.4.7-rmk3-np1-devfs	
3c589-modules-2.4.7-rmk3-np1-devfs	3c589 modules for kernel 2.4.7-rmk3-np1-devfs	
8390-modules-2.4.7-rmk3-np1-devfs	8390 modules for kernel 2.4.7-rmk3-np1-devfs	✓
ae	Anthony's Editor – a tiny full-screen editor	
aironet-modules-2.4.7-rmk3-np1-devfs	aironet modules for kernel 2.4.7-rmk3-np1-devfs	✓
apmd	Utilities for Advanced Power Management (APM) on laptops	✓
ash	NetBSD /bin/sh	✓
atd	Provides alarm services	
backpaq-modules-2.4.7-rmk3-np1-devfs	backpaq modules for kernel 2.4.7-rmk3-np1-devfs	
bash	The GNU Bourne Again SHell	✓
bkgd	Sets the X11 root window to a random image.	
blackbox	blackbox window manager	✓
bluetooth-modules-2.4.7-rmk3-np1-devfs	bluetooth modules for kernel 2.4.7-rmk3-np1-devfs	
checkers	Play checkers against the computer	
clerk	Clerk is a file manager written in python and GTK.	
cpu-scale	provides menu entries to control speed of processor	
cramfs-modules-2.4.7-rmk3-np1-devfs	cramfs modules for kernel 2.4.7-rmk3-np1-devfs	
debianutils	Miscellaneous utilities specific to Debian.	✓
dev-files	devices files for /dev directory	✓
diff	File comparison utilities	
dillo	The dillo web browser	
dosfstools	Utilities to create and check MS-DOS FAT filesystems	
e2fsprogs	The EXT2 file system utilities and libraries.	✓
ext2-modules-2.4.7-rmk3-np1-devfs	ext2 modules for kernel 2.4.7-rmk3-np1-devfs	
familiar-base	essential files for a familiar installation	✓
familiar-postinst	A collection of shell scripts to do postinstall polishing.	
fdisk	Partition a hard disk, (or other block device)	
fileutils	GNU file management utilities.	✓
fmvj18x-modules-2.4.7-rmk3-np1-devfs	fmvj18x modules for kernel 2.4.7-rmk3-np1-devfs	
foal	NeXT-style application launcher in Python	
ftp	The FTP client.	
gcardinfo	Gtk interface to basic PCMCIA card services	
gdb	The GNU Debugger	
gdk-imlib1	Gdk-Imlib is an imaging library for use with gtk	
gpsd	Provides GPS services	
grep	GNU grep, egrep and fgrep.	✓
gsoko	gSoko is GTK+ clone of Sokoban, a puzzle game where you have to push boxes over special squares.	
gtk-engines-thinice	ThinIce theme for GTK+ 1.2	
gtk-menu	A simple gtk-based menu program	
gtk-theme-thingreeber	ThinGreeber gtk theme.	
gzip	The GNU compression utility.	✓
h3600-utils	utilities for controlling hardware on Compaq iPAQ H3600 series computers	✓
hostname	A utility to set/show the host name or domain name	✓
hotplug	Hotplug add/removal scripts for H3600	✓
ide-modules-2.4.7-rmk3-np1-devfs	ide modules for kernel 2.4.7-rmk3-np1-devfs	✓
ifupdown	High level tools to configure network interfaces	✓
intimateboot	This package provides the necessary boot scripts to optionally	
ion	a text-editorish, keyboard/stylus friendly window manager	
ipaqstat	Taskbar display (ion or blackbox) date, time, apm, mem, df and wvlan	✓
ipatience	Pots o' solitaire, tiny package	

Tabelle C.1: Paketliste für Familiar 0.5 *Fortsetzung nächste Seite*

Paketname	Beschreibung	installiert
ipkg	Lightweight package management system	✓
ipkg-build	Script for building .ipk packages for the ipkg system.	
ipkg-conf	ipkg config file for the Familiar stable package feed.	
irda-common	IrDA management utilities	
irda-modules-2.4.7-rmk3-np1-devfs	irda modules for kernel 2.4.7-rmk3-np1-devfs	
jove	an emacs-like text-mode editor	
kernel-2.4.7-rmk3-np1-devfs	kernel for kernel 2.4.7-rmk3-np1-devfs	✓
kernel-modules-2.4.7-rmk3-np1-devfs	kernel modules for kernel 2.4.7-rmk3-np1-devfs	✓
labyrinth	roll a marble around (no maze yet).	
less	A file pager program, similar to more(1)	✓
libc6	GNU C Library: Shared libraries	✓
libdb2	The Berkeley database routines (run-time files).	✓
libfltk1	The Fast Light Toolkit, a GUI toolkit inspired by libForms	
libfreetype6	FreeType 2 font engine, shared library files.	✓
libglade0	Library to load .glade files at runtime.	
libglib1.2	The GLib library of C routines	
libgpmg1	General Purpose Mouse Library [libc6]	✓
libgtk1.2	The GIMP Toolkit set of widgets for X	
libjpeg62	The Independent JPEG Group's JPEG runtime library [libc6]	
libncurses5	Shared libraries for terminal handling	✓
libpam0g	Pluggable Authentication Modules library	✓
libpam-modules	Pluggable Authentication Modules for PAM	✓
libpam-runtime	Runtime support for the PAM library	✓
libpng2	PNG library & runtime	
libpopt0	lib for parsing cmdline parameters	
libreadline4	GNU readline and history libraries, run-time libraries.	
libssl0.9.6	SSL shared libraries	✓
libstdc++2.10-glibc2.2	The GNU stdc++ library	✓
libungif3g	shared library for GIF images (runtime lib)	
libwrap0	Wietse Venema's TCP wrappers library	✓
libxaw6	X Athena widget set library (version 6)	✓
libxaw7	X Athena widget set library	✓
libxft	libxft library for new font handling for X: can fail over to use	✓
libxml1	GNOME XML library	
libxmltok1	XML Parser Toolkit, runtime libraries	
libxrender	X rendering extension; needed for antialiased and alpha channel	✓
libxt	Xt Toolkit Intrinsics	✓
loadmeter	graphical CPU load montitoring	✓
login	System login tools	✓
loopback-modules-2.4.7-rmk3-np1-devfs	loopback modules for kernel 2.4.7-rmk3-np1-devfs	
lrzsz	Tools for zmodem/xmodem/ymodem file transfer	✓
madplay	Fast, high-quality, fixed-point MP3 player	✓
madplay-bleed	mp3 player (latest version) and madtime song length utility	
mbd-modules-2.4.7-rmk3-np1-devfs	mbd modules for kernel 2.4.7-rmk3-np1-devfs	
menu-tiny	dynamic menu system for packages	✓
metaprof	Profile Manager for sysset. Allows for different sysset settings for different environments.	
mingle	Mingle contact manager illustrating the use of the Familiar Framework project.	
mixer	xlib audio mixer based on sources from	✓
modutils	Linux module utilities.	✓
mount	Tools for mounting and manipulating filesystems.	✓
mtd-utils	Utilities for MTD (Memory Technology Devices.	
ncurses-base	Descriptions of common terminal types	✓
netbase	Basic TCP/IP networking system	✓
netcat	TCP/IP swiss army knife	
netkit-ping	The ping utility from netkit	✓
netlink-modules-2.4.7-rmk3-np1-devfs	netlink modules for kernel 2.4.7-rmk3-np1-devfs	✓
net-tools	The NET-3 networking toolkit	✓
newtonkbd-modules-2.4.7-rmk3-np1-devfs	newtonkbd modules for kernel 2.4.7-rmk3-np1-devfs	

Tabelle C.1: Paketliste für Familiar 0.5 *Fortsetzung nächste Seite*

Paketname	Beschreibung	installiert
nfs-common	NFS support files common to client and server	
nfsd-modules-2.4.7-rmk3-np1-devfs	nfsd modules for kernel 2.4.7-rmk3-np1-devfs	✓
nfs-kernel-server	Kernel NFS server support	
nfs-modules-2.4.7-rmk3-np1-devfs	nfs modules for kernel 2.4.7-rmk3-np1-devfs	
nls-base-modules-2.4.7-rmk3-np1-devfs	nls-base modules for kernel 2.4.7-rmk3-np1-devfs	
nls-big5-modules-2.4.7-rmk3-np1-devfs	nls-big5 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp1251-modules-2.4.7-rmk3-np1-devfs	nls-cp1251 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp1255-modules-2.4.7-rmk3-np1-devfs	nls-cp1255 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp437-modules-2.4.7-rmk3-np1-devfs	nls-cp437 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp737-modules-2.4.7-rmk3-np1-devfs	nls-cp737 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp775-modules-2.4.7-rmk3-np1-devfs	nls-cp775 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp850-modules-2.4.7-rmk3-np1-devfs	nls-cp850 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp852-modules-2.4.7-rmk3-np1-devfs	nls-cp852 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp855-modules-2.4.7-rmk3-np1-devfs	nls-cp855 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp857-modules-2.4.7-rmk3-np1-devfs	nls-cp857 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp860-modules-2.4.7-rmk3-np1-devfs	nls-cp860 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp861-modules-2.4.7-rmk3-np1-devfs	nls-cp861 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp862-modules-2.4.7-rmk3-np1-devfs	nls-cp862 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp863-modules-2.4.7-rmk3-np1-devfs	nls-cp863 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp864-modules-2.4.7-rmk3-np1-devfs	nls-cp864 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp865-modules-2.4.7-rmk3-np1-devfs	nls-cp865 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp866-modules-2.4.7-rmk3-np1-devfs	nls-cp866 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp869-modules-2.4.7-rmk3-np1-devfs	nls-cp869 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp874-modules-2.4.7-rmk3-np1-devfs	nls-cp874 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp932-modules-2.4.7-rmk3-np1-devfs	nls-cp932 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp936-modules-2.4.7-rmk3-np1-devfs	nls-cp936 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp949-modules-2.4.7-rmk3-np1-devfs	nls-cp949 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-cp950-modules-2.4.7-rmk3-np1-devfs	nls-cp950 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-euc-jp-modules-2.4.7-rmk3-np1-devfs	nls-euc-jp modules for kernel 2.4.7-rmk3-np1-devfs	
nls-euc-kr-modules-2.4.7-rmk3-np1-devfs	nls-euc-kr modules for kernel 2.4.7-rmk3-np1-devfs	
nls-gb2312-modules-2.4.7-rmk3-np1-devfs	nls-gb2312 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-13-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-13 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-14-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-14 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-15-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-15 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-1-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-1 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-2-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-2 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-3-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-3 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-4-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-4 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-5-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-5 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-6-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-6 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-7-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-7 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-8-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-8 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-iso8859-9-modules-2.4.7-rmk3-np1-devfs	nls-iso8859-9 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-koi8-r-modules-2.4.7-rmk3-np1-devfs	nls-koi8-r modules for kernel 2.4.7-rmk3-np1-devfs	
nls-koi8-ru-modules-2.4.7-rmk3-np1-devfs	nls-koi8-ru modules for kernel 2.4.7-rmk3-np1-devfs	
nls-koi8-u-modules-2.4.7-rmk3-np1-devfs	nls-koi8-u modules for kernel 2.4.7-rmk3-np1-devfs	
nls-modules-2.4.7-rmk3-np1-devfs	nls modules for kernel 2.4.7-rmk3-np1-devfs	
nls-sjis-modules-2.4.7-rmk3-np1-devfs	nls-sjis modules for kernel 2.4.7-rmk3-np1-devfs	
nls-tis-620-modules-2.4.7-rmk3-np1-devfs	nls-tis-620 modules for kernel 2.4.7-rmk3-np1-devfs	
nls-utf8-modules-2.4.7-rmk3-np1-devfs	nls-utf8 modules for kernel 2.4.7-rmk3-np1-devfs	
nmclan-modules-2.4.7-rmk3-np1-devfs	nmclan modules for kernel 2.4.7-rmk3-np1-devfs	
notepaq	Simple persistent notepad program	
ntpdate	The ntpdate client for setting system time from NTP servers.	✓
orinoco-modules-2.4.7-rmk3-np1-devfs	orinoco modules for kernel 2.4.7-rmk3-np1-devfs	✓
packet-modules-2.4.7-rmk3-np1-devfs	packet modules for kernel 2.4.7-rmk3-np1-devfs	✓
pcmcia-cs	PCMCIA Card Services for Linux.	✓
pcmcia-modules-2.4.7-rmk3-np1-devfs	pcmcia modules for kernel 2.4.7-rmk3-np1-devfs	✓
pcnet-modules-2.4.7-rmk3-np1-devfs	pcnet modules for kernel 2.4.7-rmk3-np1-devfs	✓
pikpak	Basic python-based gui frontend to ipkg	
portmap	The RPC portmapper	

Tabelle C.1: Paketliste für Familiar 0.5 *Fortsetzung nächste Seite*

Paketname	Beschreibung	installiert
ppp	Point-to-Point Protocol (PPP) daemon.	✓
ppp-modules-2.4.7-rmk3-np1-devfs	ppp modules for kernel 2.4.7-rmk3-np1-devfs	✓
procps	The /proc file system utilities.	✓
pump	Simple DHCP/BOOTP client.	✓
pyditor	python-based text editor	
pygps	gps receiver. Shows satellite positions, amplitudues,	
pygtk	a python wrapper for gtk support	
pypaq	pypaq is a madplay mp3-frontend written in python	
python	python of course ¡needs real description¿	✓
qiv	a quick image viewer for X	
reflash	reflash allows flash to be rewritten from linux while flash file	
reiserfs-module	Reiserfs is a journaling filesystem. This is the kernel module for it.	
reiserfs-modules-2.4.7-rmk3-np1-devfs	reiserfs modules for kernel 2.4.7-rmk3-np1-devfs	
rsync	fast remote file copy program (like rcp)	
ruby	An interpreter of object-oriented scripting language Ruby.	
ruby-dev	An interpreter of object-oriented scripting language Ruby.	
ruby-doc	An interpreter of object-oriented scripting language Ruby.	
ruby-tk	Ruby – the Tk bindings	
rxvt	VT102 terminal emulator for the X Window System	✓
rxvt-aa	rxvt with support for anti-aliased fonts	
sambaclient	The client version of Samba for the iPAQ	
sambaserver	The server portion of samba for the iPAQ	
screen	Console manager	
script-test	Empty test for ipkg script support	
sed	The GNU sed stream editor.	✓
serial-modules-2.4.7-rmk3-np1-devfs	serial modules for kernel 2.4.7-rmk3-np1-devfs	
shellutils	The GNU shell programming utilities.	✓
slang1	The S-Lang programming library & runtime version.	
smbfs-modules-2.4.7-rmk3-np1-devfs	smbfs modules for kernel 2.4.7-rmk3-np1-devfs	
smc91c92-modules-2.4.7-rmk3-np1-devfs	smc91c92 modules for kernel 2.4.7-rmk3-np1-devfs	
sound-modules-2.4.7-rmk3-np1-devfs	sound modules for kernel 2.4.7-rmk3-np1-devfs	✓
ssh	Secure rlogin/rsh/rcp replacement (OpenSSH)	✓
start-stop-daemon	sbin/start-stop-daemon from the dpkg package.	
stowaway-h3600	Support for the iPAQ H3600 series Stowaway keyboard	
stowaway-modules-2.4.7-rmk3-np1-devfs	stowaway modules for kernel 2.4.7-rmk3-np1-devfs	
syslog	System logging (syslog) functions	
sysset	python utility for configuring system settings	
sysvinit	System-V like init.	✓
tar	GNU tar	✓
task-backpaq-modules-2.4.7-rmk3-np1-devfs	kernel and modules required to run the Backpaq	
task-bootstrap-2.4.7-rmk3-np1-devfs	kernel and modules required to bootstrap and install more packages	✓
task-complete	An easy shortcut to give Familiar a usable environment.	
task-essential-to-boot	Task package for a minimal system that will boot, (but not much else)	✓
task-games	All of Familiar's games.	
task-intimate-modules-2.4.7-rmk3-np1-devfs	kernel and modules required to run Intimate	
task-mp3-player	Task package for a complete familiar system	✓
task-wireless	Task package for wireless networking	✓
task-x	Task package for a basic X system.	✓
tcl	The Tool Command Language (TCL)	
tcl-dev	The Tool Command Language (TCL) – the development files	
tcl-doc	The Tool Command Language (TCL)	
tcl-encoding	The Tool Command Language (TCL) – the encoding files	
textutils	The GNU text file processing utilities.	✓
tk	The Tk toolkit for tcl	
tk-dev	The Tk toolkit for tcl – the development files	
tk-doc	The Tk toolkit for tcl	
toad	Persistent to-do list manager	
tone	Musical Instrument Tuner	
util-linux	Miscellaneous system utilities.	✓
vfat-modules-2.4.7-rmk3-np1-devfs	vfat modules for kernel 2.4.7-rmk3-np1-devfs	✓

Tabelle C.1: Paketliste für Familiar 0.5 *Fortsetzung nächste Seite*

Paketname	Beschreibung	installiert
vic	Video conferencing tool with ipv6 support	
vim	Vi IMproved & enhanced vi editor	✓
wget	utility to retrieve files from the WWW via HTTP and FTP	✓
wireless-tools	Tools for manipulating Linux Wireless Extensions	✓
xaux-clients	miscellaneous X clients	
xbase-clients	Basic X clients	✓
xcalibrate	calibrate the iPAQ H3600 touch screen for X	✓
xfonts-75dpi	75 dpi fonts for X	✓
xfonts-base	standard fonts for X	✓
xfonts-ttf	free TrueType fonts to be used within X	✓
xirc2ps-modules-2.4.7-rmk3-np1-devfs	xirc2ps modules for kernel 2.4.7-rmk3-np1-devfs	
xkbd	xkbd is a small simple xlib based onscreen keyboard	✓
xlibs	X Window System client libraries	✓
xserver-tiny-h3600	X server for the iPAQ H3xxx display, touchscreen, and	✓
xstella	xstella is an Atari 2600 emulator. This package contains a patched verion to reduce the window size for ipaq use. The package also includes a python-gtk launcher and the ROMS included with the distrubution location in `/usr/share/xstella/`	
xstroke	Full-screen gesture recognition program.	✓
xstroke-help	Help mode for xstroke.	
z	A tar front end written by Steve Kinzler, `http://www.cs.indiana.edu/~kinzler`.	
zlib1g	compression library & runtime	✓

Tabelle C.1: Tabelle C.1: Paketliste für Familiar 0.5

Anhang D

SavaJe XE

Die Installation der Evaluations Version von SavaJe XE Version 0.1.2 auf dem iPAQ verläuft der Installation von Familiar sehr ähnlich. Zur besseren Übersicht werden hier wiederum alle nötigen Punkte aufgeführt. Sollte die Installation noch das Löschen von WindowsCE beinhalten, so hierzu sollte Anhang C konsultiert werden.

D.1 Bootloader

SavaJe XE bringt einen eigenen Bootloader mit, der auf den bei Familiar und LISA mLinux verwendetem, basiert, was im Umkehrschlus heißt, daß auch diese Bootloader funktionieren sollten. Wenn dennoch der bei SavaJe XE mitgelieferte zum Einsatz kommen soll, dann muß der momentane Bootloader durch den von SavaJe XE verwendeten ersetzt werden.

Hinweis: Das Ersetzen des Bootloaders ist mit einem Risiko verbunden. Sollte aus irgendwelchen Gründen etwas schieflaufen, so kann es unter Umständen vorkommen, daß der iPAQ funktionsuntüchtig wird.

Geladen wird der Bootloader folgendermaßen:

- Am `boot>`-Prompt `load bootldr` eingeben.

- Den Bootloader `bootldr-step2-0000-2.14.14-bigkernel-mono-savaJe2` via XModem-Transfer hochladen.
- Sollte das Laden erfolgreich beendet worden sein, den iPAQ neustarten.

D.2 Partionieren und Laden der Partitionen

Bei SavaJe XE liegen die nötigen Daten auf verschiedenen Partitionen. Da jedoch nicht die Standardkonfiguration verwendet wird, müssen die Partitionen entsrpechend folgender Anleitung angelegt und mit Daten gefüllt werden.

Folgende Befehle am `boot>`-Prompt eingeben:

- `partition reset`
- `partition define kernel 0x00080000 0x00380000 0`
- `partition define rt.jar 0x00400000 0x00680000 1`
- `partition define root 0x00A80000 0 8`
- `partition save`

Um zu Überprüfen ob alles richtig eingegeben und erstellt wurde, kann der Befehl `partitions show` verwendet werden. Als nächstes werden die Partitionen mit Inhalt gefüllt. Das geschieht wie üblich via XModem-Transfer.

- Kernel-Partition: `load kernel`
  Als Datei `jbitsy.bin.gz` verwenden.
- rt.jar-Partition: `load rt.jar`
  Als Datei `rt-bitsy.jar` verwenden (Hinweis: Dies kann bis zu 40 Minuten dauern).
- Abschließend die root-Partition: `load root`
  Als Datei `savaJe.romfs` verwenden.

- Zur Sicherheit sollten eventuell vorher definierte Parameter zurückgesetzt werden. Dies geschieht mit dem Befehl *params reset.*

Nun kann der iPAQ, ein erfolgreiches Laden vorrausgesetzt, neu gestartet werden und sollte SavaJe XE booten.

Anhang E

Restauration WindowsCE

In diesem Kapitel soll beschrieben werden, wie ein zuvor gesichertes WindowsCE wieder auf den iPAQ gespeichert werden kann. Der Autor möchte darauf hinweise, daß er diese Anleitung als „as is“ von Handheld übernommen hat und selbst nie eine Restaurierung von WindowsCE auf dem ihm zur Verfügung stehendem iPAQ durchgeführt hat. [18]

E.1 Vorraussetzungen

Vorraussetzung für eine Restauration von WindowsCE ist selbstverständlich eine Sicherheitskopie, die *vor* dem Installieren von mLinux, Familiar oder SavaJe XE durchgeführt wurde. In entsprechender Installationsanleitung wurde die Erstellung dieser Sicherheitskopie beschrieben. Die vier in diesem Schritt angefallenen Dateien werden für die Restaurierung benötigt:

- `flash_00000000.bin`
- `flash_00400000.bin`
- `flash_00800000.bin`
- `flash_00c00000.bin`

E.2 Restauration von WindowsCE (neue Methode)

E.2.1 Durchführung

Mit der Einführung des neuen Bootloaders 2.16.19 gibt es eine neue, einfache Methode, um WindowsCE auf dem iPAQ zu restaurieren. Führen Sie folgende Schritte durch:

- Am `boot>`-Prompt `load flash 0x00c00000` eingeben und die Datei `flash_0x00c00000.bin` via XModem-Transfer übertragen.
- Am `boot>`-Prompt `load flash 0x00800000` eingeben und die Datei `flash_0x00800000.bin` via XModem-Transfer übertragen.
- Am `boot>`-Prompt `load flash 0x00400000` eingeben und die Datei `flash_0x00400000.bin` via XModem-Transfer übertragen.
- Die letzte Datei, die auch den original Bootloader beinhaltet, muß vor dem Transfer bearbeitet werden. Am einfachsten geschieht das auf einem Linux-PC mit folgender Befehlszeile:
  `dd if=flash_00000000.bin of=flash_00040000.bin bs=1k skip=256`
  Die neu entstandene Datei `flash_00040000.bin` sollte 3 932 160 Byte groß sein. Sie kann dann anschließend mit `load flash 0x00040000` transferiert werden.
- Anschließend kann mit *load wince* der iPAQ dazu veranlaßt werden, WindowsCE zu booten. Dies ist auch die Standardeinstellung. Soll nicht WindowsCE gebootet werden, erreicht man dies durch Drücken und Halten der Mitte des Joypads nach einem Reset.

E.2.2 Überprüfung der Restauration

Nach dem Boot von WindowsCE kann auf folgende Weise festgestellt werden, ob die Restauration erfolgreich war:

- Die `Q` Taste drücken.
- Die *QUtilities* auswählen.
- Den Selbsttest auswählen.
- *ROM Checksumme* aktivieren.
- *weiter* auswählen.
- *Führe automatischen Neustart durch* aktivieren.
- Den Test mit *Start* beginnen.
- Die Testfragen beantworten.
- Überprüfen, ob der Test *ROM Checksumme* bestanden wurde.

Sollte der letzte Punkt erfüllt sein, so ist die Restauration von WindowsCE erfolgreich verlaufen.

E.2.3 Wiederherstellen des original Bootloaders

Mit einem Bootloader der Version 2.16.17 oder einer früheren Version wird bei einem Reset DRAM-Speicher gelöscht, in dem WindowsCE einige Benutzerdaten speichert. Mit der neueren Bootloadervariante sollte dieser Fehler nicht mehr auftreten. Sollte dennoch gewünscht werden, den original Bootloader wiederherzustellen, so sind folgende Schritte durchzuführen:

- **Sicherstellen, daß der iPAQ WindowsCE bootet**
- Folgenden Befehl auf einem Linux-PC ausführen:
  `dd if=flash_00000000.bin of=parrot.bin`[1] `bs=1k count=256`
- Am `boot>`-Prompt `load bootldr` eingeben.
- Die Datei `parrot.bin` via XModem-Transfer hochladen.

[1] „Parrot“ ist der Originalname des Bootloaders, der bei WindowsCE verwendet wird.

E.3 Restauration von WindowsCE (alte Methode)

Diese Methode ist durch den neueren Bootloader überholt, da sie sehr fehleranfällig ist.

Als erstes wird das *Reflash*[2]-Utility benötigt. Dieses Programm ist soll auf dem iPAQ ausgeführt werden, auf dem WindowsCE wiederhergestellt werden soll. Entpacken Sie das Archiv und wechseln Sie in das entsprechende Verzeichnis.

Vor dem „reflashen“ sollten sie so viele Prozesse beenden. Das Programm selbst wird alle Prozesse, die nicht als unproblematisch eingestuft wurden zuerst via *SIGTERM* und falls nötig via *SIGKILL* beenden. Das Reflash-Tool kann aber nur mit den original Dateinamen `flash_00[048c]00000.bin` arbeiten. Dies soll verhindern, daß aus Versehen falsche Dateien geflasht werden. Es bieten sich nun zwei Varianten:

1. **Dateien sind lokal**

   Sollten sich die Restaurationsdateien lokal zugreifbar liegen, so können Sie diese mit dem Skript *rewince* direkt benutzen. Der Befehl `./rewince.sh -d pfad-zu-den-wince-image-dateien` veranlaßt alles nötige.

   Laden Sie nie alle Image-Dateien auf den iPAQ, es sei denn, es stehen 64 MByte oder mehr zur Verfügung.

2. **Dateien sind „remote“**

   Sollten Sie nicht die Image Dateien auf dem iPAQ liegen haben, dann können Sie die bei entsprechender Gegebenheit via ZModem-Transfer dem Skript zur Verfügung stellen. Starten Sie das Skript mit dem Aufruf `./rewince-z.sh`.

Das Skript bereitet nun eine Sequenz von Befehlen vor, die für das Umflashen nötig sind. Diese Sequenz wird aufgelistet und eine Bestätigung durch den Benutzer angefordert. Bis zu dieser Stelle ist noch nichts am Flashspeicher geändert

[2] erhältlich unter ftp://ftp.handhelds.org/pub/linux/tools/reflash. Die letzte stabile Version ist unter dem Link ftp://ftp.handhelds.org/pub/linux/tools/reflash/reflash.tgz erhältlich.

worden. Sobald die Anforderung mit `y` bestätigt wurde, beginnt das Programm, den Flashspeicher umzuschreiben. **Unter keinen Umständen den Desktoprechner oder den iPAQ auschalten oder neustarten**. Das Löschen und Neuschreiben beansprucht etwa 5 Minuten. Nach der Programmierung wird noch eine Byteweise Überprüfung des umgeschriebenen Flashspeichers vorgenommen. Sollte alles erfolgreich verlaufen sein, so wird *Reflash* den iPAQ neu starten. Es sollte dann der WindowsCE Startbildschirm erscheinen. Sollte ein Fehler aufgetreten sein, so wird *Reflash* in einer Schleife die notwendigen Schritte wiederholen, bis diese Erfolgreich abgeschlossen werden. Konsultieren Sie die Datei `README.reflash` für weitere Details, was in dieser Phase getan werden kann. Sollten auch diese Hinweise keinen Erfolg bringen, so setzen Sie sich per Email mit Handhelds.org[3] in Verbindung, fügen soviel Debug-Hinweise wie möglich hinzu und belassen sowohl iPAQ als auch Desktop-PC in ihrem derzeitigen Zustand.

[3] `info@handhelds.org`

Literaturverzeichnis

[1] Giovanni Adorni, Frederico Bergenti, Agostino Poggi, and Giovanni Rimassa. *Enabling FIPA Agents on Small Devices.* available at `leap.crm-paris.com/public/docs`.

[2] David Anderson (Director) et al. *The Search for Extraterrestrial Intelligence* at Home. `http://setiathome.berkeley.edu` [Zugriff am 20.11.2001].

[3] Blackdown.org.
Homepage: `http://www.blackdown.org/`
FTP-Server: `ftp://ftp.blackdown.org` (Anonymer Zugriff möglich)
[Zugriff am 20.11.2001].

[4] Douglas McConnaughey Boling. *Programming Microsoft Windows CE.* Microsoft Press International, 2001.

[5] F. A. Brockhaus. *Der Volksbrockhaus - Jubiläumsausgabe.* F.A. Brockhaus Verlag, Wiesbaden, 1956.

[6] compaq.com. *iPAQ PocketPC compatible hardware.*
`http://www.compaq.com/products/handhelds/pocketpc/compatiblehw.html`
[Zugriff am 28.11.2001].

[7] O. Etzioni and Dan Weld. "List of qualities of software agents", in IEEE Expert, 1995.

[8] FidoNET.
`http://fidonet.fidonet.org/`
[Zugriff am 20.11.2001].

[9] David Flanagan. *Java in a Nutshell, 2nd Edition.* O'Reilly & Associates, Inc., 1997.

[10] Lonnon R. Foster. *Palm OS Programming Bible.* Hungry Minds, 2000.

[11] Rainer Gievers. *Pocket PC und Windows CE 3.0 für Einsteiger.* BoD, Norderstedtl, 2001.

[12] Clemens Gleich. *Gleichschaltung - Privates Parrallelrechnen im Internet.* c't Magazin für Computer Technik 22/01 Seite 202, Heise Verlag.

[13] Li Gong. *Inside Java 2 Platform Security.* Addison Wesley, 1999.

[14] James Gosling, Bill Joy, and Guy L. Steele. *The Java Language Specification.* Addison Wesley, 1996.

[15] Alexander Guy and Ken Causey. *Familiar 0.5 *Pre-Release* Installation Guide.* `http://familiar.handhelds.org/familiar/feeds/unstable/install/H3600/install.html`
[Zugriff am 28.11.2001].

[16] Handhelds.org. FTP-Server.
`ftp://familiar.handhelds.org/` (Anonymer Zugriff möglich)
[Zugriff am 28.11.2001].

[17] Handhelds.org. *The Familiar Project.*
`http://familiar.handhelds.org/`
[Zugriff am 28.11.2002].

[18] Handhelds.org. *WinCE Restauration.*
`http://handhelds.org/projects/wincerestoration.html`
[Zugriff am 9.11.2001].

[19] Markus Holtmann. *Wireless LAN und Linux.*
`http://www.holtmann.org/linux/wlan/`
[Zugriff am 2.2.2002].

[20] Thomas Hottum. *Nachrichtentransport auf Basis Mobiler Agenten.* Diplomarbeit, Fachhochschule Bingen, 2002.

[21] IBM Visual Age.
`http://www.embedded.oti.com/`
`http://www.ibm.com/software/ad/embedded`
[Zugriff am 06.12.2001].

[22] IBM Visual Age Micro Edition. *Frequently Asked Questions.*
`http://www.embedded.oti.com/learn/faq_crt.phtml`
[Zugriff am 03.2.2001].

[23] IKV++. Grasshopper 2-Homepage.
`http://www.grasshopper.de/`
[Zugriff am 20.11.2001].

[24] JADE - Java Agent DEvelopment Framework - Homepage.
`http://sharon.cselt.it/projects/jade/home.htm`
[Zugriff am 11.12.2001].

[25] javamobiles.com. *List of JVM for PDA.*
`http://www.javamobiles.com/jvm.html`
[Zugriff am 28.11.2001].

[26] javazoom. *Java MP3 Player.*
`http://www.javazoom.net/javalayer/javalayer.html`
[Zugriff am 26.1.2001].

[27] Kada Systems. *Developer's Guide.*
Enthalten in der KadaVM Distribution im `docs`-Verzeichnis.

[28] Kada Systems. *KadaVM Homepage.*
`http://www.kadasystems.com`
[Zugriff am 28.11.2001].

[29] Martin Kahmann. *Report Mobile Business.* Symposium Publishing, 2001.

[30] Marc Kersten. *ARM in ARM - Das Ende der Prozessorvielfalt bei PDAs ?* c't Magazin für Computer Technik 17/01, Heise Verlag.

[31] Michael Kofler. *Linux - Installation, Konfiguration, Anwendung.* Addison-Wesley-Longman, 1999.

[32] Helmut Kopka. *LaTeX Band 1 — Einführung.* Addison-Wesley (Deutschland GmbH, Bonn, 1996. 2. überabreitete Auflage.

[33] Helmut Kopka. *LaTeX Band 2 — Ergänzungen. Mit einer Einführung in Metafont.* Addison-Wesley (Deutschland GmbH), Bonn, 1997. 2. überabreitete Auflage.

[34] Doug Lea. *Concurrent Programming in Java. Entwurfsprinzipien und Muster.* Addison Wesley, Bonn, 1997.

[35] LEAP-Homepage.
`http://leap.crm-paris.com/`
[Zugriff am 11.12.2001].

[36] Robert Leslie. *MAD: MPEG Audio Decoder.*
`http://www.mars.org/home/rob/proj/mpeg/`
[Zugriff am 26.1.2001].

[37] Sheng Liang and Gilad Bracha. *Dynamic Class Loading in the Java(TM) Virtual Machine.*

[38] Tom Lindholm and Frank Yellin. *The Java Virtual Machine Specification.* Addison Wesley, 1996.

[39] LISA. FTP-Server.
`ftp://ftp.lisa.de/pub/mLinux/` (Anonymer Zugriff möglich)
[Zugriff am 09.11.2001].

[40] LISA. *mLinux v0.9 Installationsanleitung.*
Auf der LISA mLinux-CD unter `mLinux-0.9/docs/manual/install.html`
[Zugriff am 09.11.2001].

[41] memorymanagement.org. *The Memory Management Glossary - Garbage Collection.*

`http://www.memorymanagement.org/glossary/g.html`
[Zugriff am 5.2.2002].

[42] Rainald Menge. *Java calling - Entwicklerkonferenz JavaOne in San Francisco.* c't Magazin für Computer Technik 14/99, Heise Verlag.

[43] Michel Goossens, Frank Mittelbach, Alexander Samarin. *Der LaTeX-Begleiter.* Addison-Wesley (Deutschland GmbH), Bonn, 1998. 4. unveränderter Nachdruck der 1. Auflage.

[44] Sun Microsystems. *Java JIT Compiler Overview.*
`http://www.sun.com/solaris/jit/`
[Zugriff am 1.2.2002].

[45] Sun Microsystems. *Java(TM) Sound API Programmer's Guide.*
`http://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/title.fm.html`.

[46] mkfs.jffs2. verfügbar unter
`ftp://ftp.uk.linux.org/pub/people/dwmw2/iPAQ/mkfs.jffs2` (Anonymer Zugriff möglich)
[Zugriff am 24.10.2001].

[47] Peter Monta. *Installation Without Windows.* Handhelds.org Online Forum, 2000.
`http://www.handhelds.org/pipermail/ipaq/2000-August/000061.html`
[Zugriff am 09.11.2001].

[48] K.W. Ng. *Mobile Agent Technology.* Department of Computer Science & Engineering - The Chinese University of Hong Kong
`www.cse.cuhk.edu.hk/~kwng/7250/ma313.pdf`.

[49] O'Reilly & Associates. *XModem.*
`http://www.oreilly.com/reference/dictionary/terms/X/XModem.htm`
[Zugriff am 2.2.2001].

[50] PDA Wear.com. *Compaq Ipaq H3650.*
`http://www.pdawear.com/pdawalk/pdaw_compaq_ipaqh3650.htm`
[Zugriff am 7.11.2001].

[51] PDA Wear.com. *Palm V Specification.*
`http://www.pdawear.com/pdawalk/pdaw_palmv.htm`
[Zugriff am 7.11.2001].

[52] Ulrich Pinsdorf. *Entwicklung eines Dialogsystems zur multimodalen Interaktion zwischen Mensch und Agenten unter besonderer Berücksichtigung der natürlichen Sprache.* Fachhochschule Bingen, 1999.

[53] PKWARE Inc. *PKWARE Releases New ZIP File Format Specification.*
`http://www.pkware.com/abouts/news/pr_20011205.html`
[Zugriff am 5.2.2002].

[54] Peter Rausch. *Software Engineering. Skript zur Vorlesung "Software Engineering".* Fachhochschule Bingen, 1995.

[55] Peter Rausch. *Objektorientierte Systementwicklung mit der Unified Modeling Language (UML). Skript zur Vorlesung "Objektorientierte Systementwicklung".* Fachhochschule Bingen, 2000.

[56] Jürgen Rink and Thomas Schult. *Dicke Zwerge - PDAs mit Windows CE im Vergleich.* c't Magazin für Computer Technik 08/98 Seite 124, Heise Verlag.

[57] Volker Roth. *Security in SeMoA Version 2: An overview.* Im Installationsverzeichnis von SeMoA unter `docs`.

[58] Volker Roth, Ulrich Pinsdorf, et al. *SeMoA: Dokumentation zur SeMoA Plattform.* Im Installationsverzeichnis von SeMoA unter `docs`.

[59] SavaJe Technologies.
`http://www.savaje.com/`
[Zugriff am 23.11.2001].

[60] Socket Communications. *What is an ESSID.*
`http://www.socketcom.com/support/faq_wlan_7.htm`
[Zugriff am 2.2.2002].

[61] Insignia Solutions. *Jeode Homepage.*
`http://www.insignia.com/java_enabled.htm`
[Zugriff am 24.1.2002].

[62] Andreas Stiller. *Die ARM Story - Von der kleinen Architektur zum großem Marktführer.* c't Magazin für Computer Technik 02/02 Seite 70, Heise Verlag.

[63] Sun Microsystems. *Java2(TM) Platform.*
`http://java.sun.com/java2`
[Zugriff am 28.11.2001].

[64] Sun Microsystems. *Java2(TM) Platform, Micro Edition.*
`http://java.sun.com/java2me`
[Zugriff am 28.11.2001].

[65] Sun Microsystems. *Java(TM)2 Platform Security Introduced.*
`http://java.sun.com/j2se/1.3/docs/guide/security/index.html`
[Zugriff am 2.2.2002].

[66] Sun Microsystems. *Java(TM) 2 Platform Micro Edition (J2ME(TM)) Technology for Creating Mobile Devices, White Paper.* published over web, 2000.
`http://java.sun.com/java2me`
[Zugriff am 28.11.2001].

[67] Sun Microsystems. *Connected Device Configuration (CDC) and the Foundation Profile - Technical White Paper.* published over web, 2001.
`http://java.sun.com/java2me`
[Zugriff am 28.11.2001].

[68] Sun Microsystems. *MIDP APIs for Wireless Applications - A Brief Tour for Software Developers - A White Paper.* published over web, 2001.
`http://java.sun.com/java2me`
[Zugriff am 28.11.2001].

[69] Karsten Violka. *Java für Handy - Mobile Anwendungen selbst entwickeln.* c't Magazin für Computer Technik 21/01 Seite 266, Heise Verlag.

[70] Karsten Violka. *Kaffee unterwegs - Mobile Anwendungen mit der Java Micro Edition entwickeln.* c't Magazin für Computer Technik 01/02 Seite 184, Heise Verlag.

[71] Wabasoft. *Waba Homepage.*
`http://wabasoft.com`
[Zugriff am 1.2.2002].

[72] Tom Wilkinson. *Welcome to Kaffe.*
`http://www.kaffe.org/`
[Zugriff am 23.11.2001].

[73] Dusan Zivadinovic. *Einer für alles - Smartphones vereinen Mobiltelefon und Handheld* und *kommunikationsassistenten - sieben smartphones im vergleich.* c't Magazin für Computer Technik 14/01 Seite 90, Heise Verlag.

[74] Volker Zota. *Westentaschen-Pinguin - Linux auf aktuellen PDAs.* c't Magazin für Computer Technik 03/01 Seite 136, Heise Verlag.

Selbständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum

Rolf Grindel

Realisierung einer sicheren Mobilen Agenten Plattform auf einem PDA

Diplomarbeit WS 2001/2002

Rolf Grindel