Die JavaCard als Programmier- und Ausführungsplattform für verteilte Anwendungen[1]

Stefan Fünfrocken[2], Friedemann Mattern[3], Marie-Luise Moschgath[4]

[2] TU Darmstadt, fuenf@informatik.tu-darmstadt.de
[3] ETH Zürich, mattern@inf.ethz.ch
[4] TU Darmstadt, moschgath@ito.tu-darmstadt.de

Zusammenfassung: Chipkarten, die durch ihren integrierten Prozessor einfach zu nutzende Rechenkapazität an einem beliebigen Ort bereitstellen, finden in immer mehr Bereichen Verwendung. In jüngster Zeit wurden Kartenbetriebssysteme entwickelt, die Mehrfachanwendungen ermöglichen, sowie Karten, die in einer Hochsprache programmiert werden können und die es erlauben, Applikationen dynamisch auf die Karte nachzuladen. Eine besondere Rolle spielen hier Java-basierte Chipkarten. Mit dieser sogenannten JavaCard gelingt die Einbindung in offene bzw. netzbasierte Anwendungsszenarien besonders leicht, da Java-basierte Ausführungsumgebungen im Internet und in Intranets weit verbreitet sind. Die JavaCard stellt als Programmier- und Ausführungsplattform somit eine „portable" Komponente in einem verteilten Anwendungsverbund dar, wobei durch das Schutzkonzept von Java die Sicherheit der Kartenapplikationen zusätzlich erhöht wird.
Der Beitrag skizziert prototypisch realisierte Anwendungsszenarien mit CORBA-Anbindung der JavaCard und beschreibt die daraus gewonnenen Erfahrungen und das Entwicklungspotential für Java-fähige Chipkarten in netzbasierten Applikationen.

1 Einleitung

Als vor knapp 30 Jahren mehrere Patente zur Einbringung einer Prozessorschaltung in eine Plastikkarte angemeldet wurden, ahnte noch niemand, wie vielseitig die Anwendungen von Chipkarten in Zukunft sein würden. In der Zwischenzeit entwickelte sich die Chipkarte, weitgehend unbemerkt von der Informatik-Fachwelt, vom reinen Datenspeicher (Speicherchipkarten) über einfache Spezialprozessoren (Mikrocontrollerkarten) hin zu vollwertigen, universellen Prozessoren - quasi Computer im Kleinstformat für die Brieftasche oder Geldbörse - mit derzeit ca. 32 kB Speicher, einer 32-Bit RISC-Architektur und einigen MFLOPs Rechenleistung [1]. Damit entsprechen die Chipkarten leistungsmäßig den PC-Prozessoren vor einigen Jahren; ihnen fehlt zum vollwertigen Rechner eigentlich nur die Stromversorgung sowie die eigene Eingabe- und Ausgabemöglichkeit, beides wird von den Kartenterminals bereitgestellt.

[1] Diese Arbeit wurde vom Zentrum für Kartenanwendungen sowie dem Technologiezentrum Darmstadt der Deutschen Telekom AG gefördert.

Chipkarten erlangten in letzter Zeit aufgrund ihrer einfachen, praktischen Handhabbarkeit sowie ihrer vielseitigen Verwendbarkeit eine starke Verbreitung, insbesondere im Bereich sicherheitsrelevanter Applikationen: Logische und physikalische Schutzmechanismen verhindern unautorisierte Zugriffe auf die in den Chips gespeicherten Daten oder geheimen Schlüssel, außerdem erlauben integrierte Krypto-Coprozessoren schnelle kryptographische Berechnungen. Chipkarten sind somit hervorragend als „portables“ Identifikations- und Authentifizierungsmedium geeignet. Damit stellt die Chipkarte eine wichtige technische Komponente für Anwendungen im Bereich digitale Signaturen, Electronic Commerce und Electronic Banking dar.

Gegenwärtig sind bereits mehrere hundert Millionen Chipkarten in Umlauf, wobei allgemein erwartet wird, daß die derzeitige Steigerungsrate auch in den kommenden Jahren unvermindert anhalten wird: Kartenterminals sind mittlerweile als flächendeckende Infrastruktur vorhanden (Geldautomaten, öffentliche Telefone etc.), billige Lesegeräte für PCs beginnen den Massenmarkt zu erobern, erste Multifunktionsanwendungen werden im Kontext der Geldkarte erkennbar und wichtige Technologiebranchen wie beispielsweise Hersteller von Mobiltelefonen oder Anbieter von Mobilkommunikationsdiensten betreiben energisch die Weiterentwicklung der Chipkartenfunktionalität. Neue Entwicklungen wie die kontaktlose Chipkarte, die SuperSmartcard - eine Chipkarte mit integriertem Display und Eingabemöglichkeit - oder auch Hybridkarten, die verschiedene Kartentechnologien auf einer einzigen Karte vereinigen, bilden dabei die technischen Voraussetzungen für neue Einsatzmöglichkeiten und eine damit einhergehende weitere Verbreitung.

Darüber hinaus zeichnet sich derzeit ab, daß Chipkarten als flexible System- und Ausführungskomponenten eine neue, zusätzliche Rolle in vernetzten Anwendungsszenarien zukommen wird. Ansätze zur Einbindung von Chipkarten als Ausführungsplattformen in verteilte Applikationen werden im folgenden dargestellt.

2 Die JavaCard

In gleichem Maße, wie die Verbreitung und Akzeptanz von Chipkarten wächst, nehmen auch die Anforderungen zu. Komplexere Anwendungen, schneller aufeinanderfolgende Softwaregenerationen, aber auch der wachsende Konkurrenz- und Kostendruck bedingen neue Konzepte in der Chipkartenwelt. Forderungen nach nachladbarer Funktionalität und dynamisch erweiterbaren Applikationen, einfacherer und schnellerer Softwareentwicklung sowie Multiapplikationsfähigkeit ohne Kompromisse hinsichlich der Sicherheit gespeicherter Daten oder fremder Funktionseinheiten werden sowohl seitens der Chipkartenhersteller als auch der Kartenemittenten laut.

Entsprechend den technischen Gegebenheiten entwickelte sich in den letzten Jahren auch die Softwareseite weiter: Analog zur „Großrechnergeschichte“ vor ca. 25 Jahren findet gegenwärtig der Übergang von der Chipkartenprogrammierung in Maschinensprache zur Verwendung von Hochsprachen (wie beispielsweise Java, C oder Basic) statt, mit z.T. um den Faktor zehn verkürzten Entwicklungs- und Projektdurchführungszeiten. Ferner kommen (zwar abgespeckte, aber funktional weitgehend vollwertige) Betriebssysteme zum Einsatz, mit denen multifunktionale Chipkarten ermöglicht werden, wodurch mehrere unterschiedliche Anwendungen unabhängig voneinander (bzw. in koordinierter Weise miteinander) auf einer Karte ablaufen können.

Aufgrund ihrer einfachen Einbindungsmöglichkeit in netzbasierte Anwendungsszenarien sind Chipkarten, die in Java programmiert werden können, besonders interessant. Eine JavaCard [3] ist eine Chipkarte, die neben einem beliebigen Betriebssystem einen Java-Interpreter (eine sogenannte Java Virtual Machine (VM)) enthält. Die von der Java-VM zu unterstützenden Funktionen sind im „Java Card API“ (Application Programming Interface) [3] definiert und legen i.w. die Kommunikation mit der Umgebung und den Zugriff auf Ressourcen der Karte fest.

Die Verwendung von Java in einer Chipkarte ist wegen der einfacheren und schnelleren Anwendungsentwicklung auch durch Nicht-Chipkarten-Experten von wachsender Bedeutung. Durch die Nutzung von Java wird außerdem die Interoperabilität mit externen, in Java realisierten Anwendungskomponenten unterstützt. Da Chipkarten vielfach im Kontext sicherheitsrelevanter Anwendungen eingesetzt werden (u.a. als vertrauenswürdige Ausführungsplattform), ist besonders die in gewisser Weise bereits eingebaute Sicherheit und Robustheit von Java bemerkenswert, die durch die Verwendung eines die Ausführung „kontrollierenden“ Interpreters in Verbindung mit Bytecode-Verifikation [7] erreicht wird, auch wenn im Detail einige diesbezügliche Aspekte noch Forschungsgegenstand sind [9].

Obwohl Java für eingebettete Systeme entwickelt wurde, sind die Ressourcenanforderungen bezüglich Speicherkapazität und Prozessorleistung noch zu hoch für heutige Chipkarten. Eine Einschränkung des Befehls- und Funktionsumfanges von Java war daher notwendig, um Java in Chipkarten einsetzen zu können. Aus diesem Grund wurde im Februar 1997 von den führenden Kartenherstellern das JavaCard Forum (JCF) [4] gegründet. Wichtigstes Ziel dieses Konsortiums ist die Definition des Java Card API.

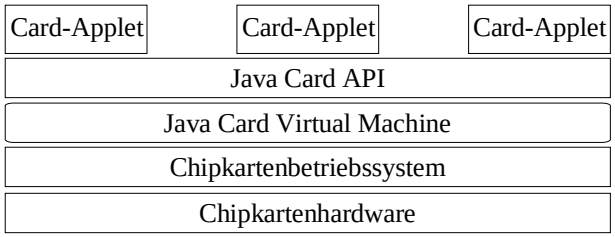

Abbildung 1: JavaCard-Softwarearchitektur

Der aktuelle JavaCard 2.1-Standard definiert die Grundfunktionalität einer Java-Chipkarte (siehe **Abbildung 1**), das sogenannte JavaCard Runtime Environment. Dieses besteht aus der Java Card Virtual Machine sowie dem Java Card API. Der durch das API definierte reduzierte Sprachumfang unterstützt kein dynamisches Nachladen von Klassen, kein Multithreading, keine mehrdimensionalen Felder und auch nicht das Clonen von Objekten. Von den Datentypen werden float, double und long nicht, int wird optional von Karten mit 32-Bit-Prozessor unterstützt. Aus Kapazitätsgründen wird bei gegenwärtigen Implementierungen oft auch kein Garbage-Collection durchgeführt.

Zwar ist aufgrund der knappen Ressourcen gegenwärtiger Chipkarten der Sprachumfang reduziert, dies schränkt die Verwendung von Java für typische Applikationen i.a. jedoch nicht wesentlich ein: Abgesehen von den beschriebenen Einschränkungen

laufen Java-Programme überall dort, wo eine Java-VM vorhanden ist: In einem Server, einem PC, einem WWW-Browser oder eben einer Chipkarte. Es ist davon auszugehen, daß sich sowohl der nutzbare Speicher als auch die Prozessorleistung von Chipkarten in den nächsten Jahren noch wesentlich steigern läßt, so daß die heute noch existierenden Einschränkungen im Sprachumfang aufgehoben werden können.

3 Die JavaCard als Infrastrukturkomponente in verteilten Anwendungen

Verteilte Applikationen haben heute, wo das Internet oftmals die zugrundeliegende „Plattform“ darstellt, eine ganz andere Dimension als noch vor kurzem. Dies macht die Verwendung einer neuen Art von Middleware als Infrastruktur für solche Applikationen erforderlich, bei der auch die Chipkarte als eine Komponente, die innerhalb eines solchen Systemverbundes eine Teilfunktionalität erfüllt, unterstützt werden sollte. Gegenwärtig ist auch zu beobachten, daß das Internet nicht nur weiter stetig wächst, sondern auch dynamischer und mobiler wird, wozu technische Weiterentwicklung wie drahtlose Netze (z.B. Bluetooth [2]), höhere Kommunikationsleistung oder weitere Miniaturisierung sowie ökonomisch-politischen Gegebenheiten (Offenheit der Schnittstellen, Interoperabilität, kürzere Produktzyklen, Globalisierung der Märkte etc.) beitragen. Die Konsequenz ist, daß auch die Applikationen in einem verteilten Systemverbund offener, dynamischer und prinzipiell umfänglicher werden als traditionelle netzbasierte Anwendungen.

Als mittlerweile „klassische“ Middleware-Plattform stellt CORBA einen Infrastrukturrahmen zur Unterstützung von Applikationen bereit, welche aus ggf. weiträumig verteilten miteinander kooperierenden Objekten bestehen. Herkömmliche verteilte Architekturprinzipien sind für große, sehr dynamische und völlig offene Systeme jedoch zu wenig adaptiv und daher eher ungeeignet. Mit Java wurde nun eine Sprache konzipiert, die von ihren Konzepten her besser für das Internet und seine neuen Anwendungen geeignet ist als klassische Programmiersprachen. Allerdings gehört zu einer guten „Internet-Infrastruktur“ wesentlich mehr als eine Sprache: Gefordert sind entsprechend geeignete Ablaufumgebungen, Betriebssysteme und Verteilungsplattformen. Bezüglich Java hat Sun kürzlich mit Jini [5] ein auch für diesen Zweck hochinteressantes Framework vorgestellt.

Es sieht daher so aus, als ob man am Anfang einer neuen Generation von „Internet-Middleware“ steht, die durch ihre Mechanismen und Paradigmen die Dynamik und Offenheit der sich abzeichnenden großen verteilten Applikationen besser unterstützt. Andererseits steht ein fast alles durchdringendes Internet mit seinen vielfältigen interoperablen Diensten und der propagierten Offenheit, das teilweise auch Verantwortlichkeiten virtualisiert, zunächst im Gegensatz zu Sicherheit und Vertrauenswürdigkeit. Da sich nun gerade Chipkarten in großem Stile als Instrumente zur Erhöhung von Sicherheitsbelangen durchsetzen, ist die Frage interessant, welche Rolle Chipkarten in dieser neuen Welt spielen können. Insbesondere die JavaCard ist in dieser Hinsicht bemerkenswert, da sie Programme in einer sicheren Umgebung ausführen kann und für mobilen Code und Szenarien aus dem Bereich des Electronic Commerce [8] prädestiniert zu sein scheint. Durch ihre eingebaute virtuelle Java-Maschine und die Möglichkeit, dynamisch Programmkomponenten aus dem Netz zu laden, stellt sie

zudem (relativ zur „Java-Welt“) eine universelle, sichere Plattform in einer offenen, verteilten Welt dar.

Um Chipkarten in einem offenen Verbund mit anderen Geräten und Services betreiben zu können, wird eine Eingliederung in eine standardisierte Middleware-Architektur notwendig. Im nachfolgenden Kapitel wird daher skizziert, wie eine Anbindung der JavaCard an die CORBA-Infrastruktur realisiert werden kann. Die weitergehende Verwendung der JavaCard als sichere Ausführungsumgebung für mobilen Code wird an anderer Stelle beschrieben [6].

4 Prototypische Realisierung verteilter JavaCard-Applikationen

Um das Potential der JavaCard beim Einsatz in zukünftigen Diensten und verteilten Anwendungen aus dem Bereich der Telekommunikation zu untersuchen, wurden von uns einige Applikationsszenarien prototypisch realisiert. Beispielhaft dafür werden nachfolgend zwei davon - die CORBA-Anbindung und die Realisierung einer Lotto-Kundenkarte - beschrieben.

4.1 Middleware-Integration

Eine Middlewareplattform wie CORBA enthebt den Programmierer netzbasierter Anwendungen von vielen Details, die durch die verteilte Natur der Applikation entstehen: Das Kernstück jeder CORBA-konformen Middleware, der sogenannte Object Request Broker (ORB), übernimmt die ortstransparente Vermittlung zwischen räumlich getrennten Objekten. Dies kann man sich auch im vorliegenden Kontext zunutze machen: Ein Applet, das auf der JavaCard abläuft, benötigt einen lokalen Kommunikationspartner im Kartenterminal, der die passive Karte anspricht. Einmal angestoßen, kann es die Applikation auf der Karte erfordern, sich mit einem weiter entfernt im Netz angesiedelten Dienst zu verbindet. Das Master/Slave-Prinzip, das für Chipkarten im Standard ISO/IEC 7816 festgelegt ist, verhindert jedoch, daß die Karte von sich aus aktiv werden kann.

Um beide Probleme zu lösen, wird eine Vermittlungsinstanz im Kartenterminal benötigt, die das Master/Slave-Prinzip umgeht und eine Verbindung zu einem Netzdienst aufbauen kann. Das Master/Slave-Problem wurde entsprechend dem „SIM Toolkit“-Konzept [10] gelöst und wird zum einfacheren Verständnis in der weiteren Betrachtung vernachlässigt.

Um den Aufwand, der durch die jeweils applikationsspezifisch zu realisierende Vermittlungsinstanz entsteht, zu eliminieren, sollte die ORB-Funktionalität einem Karten-Applet direkt auf der Chipkarte zur Verfügung stehen. Für einen solchen „Card-ORB“ (CORB) bieten sich zwei Realisierungsmöglichkeiten an:

- Proxy-Lösung: Der Teil des CORB, der die Vermittlungsfunktionalität realisiert, ist auf das Kartenterminal ausgelagert (siehe **Abbildung 2**). Die Klassen, die den CORB auf der Karte selbst realisieren, sind dabei Proxy-Objekte, die die Funktionalität des extern angesiedelten CORB-Teiles in die Karte spiegeln, indem die Objektreferenzen und Dienstanfragen dorthin weitergereicht werden. Für die eigentlichen Dienstobjekte muß in gleicher Weise verfahren werden.

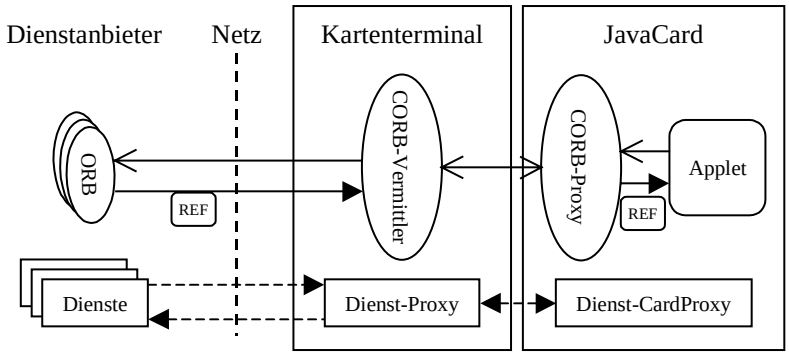

Abbildung 2: Einfaches ORB-Szenario

- Integrierte Lösung: Die Vermittlungsfunktionalität ist vollständig durch die CORB-Klassen auf der Chipkarte implementiert. Läuft die gesamte ORB-Funktionalität auf der Karte ab, so kann den Kartenapplikationen ein vertrauenswürdiger ORB angeboten werden, da die Karte als sichere Ablaufumgebung gilt. Dies erhöht die Sicherheit beim Arbeiten in unbekannter Umgebung und mit unbekannten externen Komponenten erheblich.

Die Implementation der gesamten ORB-Funktionalität auf der JavaCard im Sinne einer integrierten Lösung ist zur Zeit aus Platz- und Performanzgründen noch nicht realisierbar. Dies kann erst mit der Verfügbarkeit leistungsfähigerer Karten erreicht werden. Aus diesem Grund wurde für die Prototypapplikationen die Proxy-Lösung realisiert.

Die Vorteile einer Middleware-Integration liegen klar auf der Hand: Karten-Applets können in einfacher und flexibler Weise an jedem Kartenterminal auf beliebige Netzdienste zugreifen, indem sie alle Möglichkeiten einer CORBA-basierten Middleware-Plattform nutzen. Dadurch können einem Kunden als Nutzer der JavaCard beliebige, neuartige Dienstleistungen über seine Karte angeboten werden, gleichgültig, wo sich der Kunde gerade befindet, bzw. wo der Netzdienst angesiedelt ist.

4.2 Personalisierte Kundenkarte: Die Lotto-JavaCard

Die von uns in einem weiteren Szenario realisierte Lotto-JavaCard ist ein Beispiel für eine maßgeschneiderte Privatkundenkarte. Das realisierte Prototypszenario besteht aus den in **Abbildung 3** gezeigten Komponenten: Der Lotto-Kunde erhält eine persönliche Lotto-JavaCard, mit der er überall und zu jeder beliebigen Zeit Lotto spielen kann. Einzige Voraussetzung ist ein Telefonanschluß (Handy, Telefonzelle, einfaches Telefon oder PC mit Modemanschluß) mit integriertem Kartenlesegerät. Die Lottogesellschaft unterhält einen Netzdienst „Lottoserver“. Jedem Kunden wird eine eindeutige, aber anonyme Benutzer-ID zugeordnet; zusätzlich erhält er ein Konto, von dem die Spielkosten abgezogen und auf das Gewinne gutgeschrieben werden. Die von der Karte über das Netz erhaltenen Tipreihen eines Kunden werden unter dessen Benutzer-ID bis zur Ziehung gespeichert.

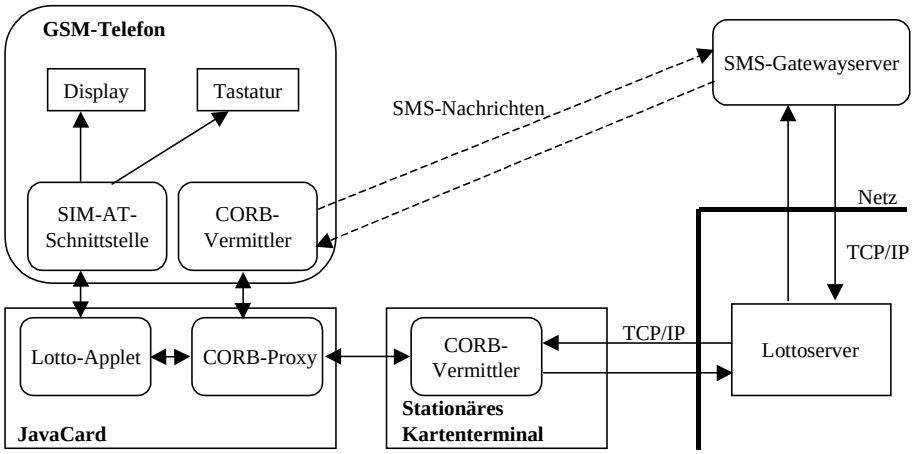

Abbildung 3: Architektur des Lotto-JavaCard-Szenarios

Das auf der Karte befindliche Lotto-Applet übernimmt dabei folgende Aufgaben:

- Ansteuerung des Displays des Kartenterminals zum Anzeigen von applikationsspezifischen Menüs.
- Verschlüsselte Kommunikation mit dem Lottoserver zur Tip-Abgabe und Gewinnüberprüfung mit gegenseitiger Authentifizierung mittels Challenge-Response-Verfahren.
- PIN-gesicherte Speicherung von Authentifizierungsdaten (Benutzer-ID und Paßwort) und Diffie-Hellman-Schlüssel zur Verschlüsselung der Kommunikation.
- Speicherung der Applikationsdaten (z.B. Tip-Reihen, Gewinnzahlen).
- Verwaltung des Lottokontos: Auf-, Ab- oder Umbuchen über eine Kreditkarte, ein Girokonto oder aber eine auf der Karte befindliche Geldbörsen-Applikation (z.B. GeldKarte, PayCard).

Das Prototypszenario wurde für den Einsatz der Karte in einem GSM-Mobiltelefon konzipiert, wobei das Mobiltelefon als Java-Applikation simuliert wurde. Um die Menüansteuerung realitätsgetreu nachzubilden, wurde eine SIM-AT-konforme [10] Java-Schnittstelle implementiert. Da ein GSM-Telefon als einzige Möglichkeit zur Kommunikation mit einem Server im Telekommunikationsnetz SMS-Messages zur Verfügung hat, wurde zusätzlich ein SMS-Gatewayserver realisiert. Dieser nimmt SMS-Messages entgegen und leitet die Daten über TCP/IP-Kommunikationsprotokolle an den Lottoserver weiter. Für die Antwortnachrichten vom Lottoserver werden die Daten vom Gatewayserver in SMS-Nachrichten umgewandelt und an die Lotto-Applikation gesendet. Wird die Karte an einem stationären Terminal mit Netzanschluß verwendet, kann die Kommunikation unmittelbar über das Netz erfolgen.

5 Eignung der JavaCard als Plattform verteilter Anwendungen

Damit Applikationen von einer verteilten Anwendungsarchitektur, in die die Java-Chipkarte eingebettet ist, profitieren können, müssen eine Reihe von infrastrukturellen Voraussetzungen erfüllt sein. Diese lassen sich in zwei Kategorien unterteilen:

1. Infrastrukturmaßnahmen auf den Karten selbst, und
2. kartenexterne Infrastrukturmaßnahmen.

In einem idealen verteilten Applikationsszenario sollte eine Chipkarte nicht anders behandelt werden als jeder andere Netzknoten, der allerdings spezielle Eigenschaften aufweist. Soll eine Chipkarte wie ein beliebiger Rechner in einem Intranet oder im Internet angesprochen werden können, so ist die Möglichkeit der direkten Kommunikation mit der Karte essentiell. Da Middleware-Realisierungen als Kommunikationsplattformen grundlegende Mechanismen zur Implementierung verteilter Anwendungen bereitstellen, liegt es nahe, die Integration von Chipkarten mit solchen Architekturen voranzutreiben, um diese so in uniformer Weise zugreifbar und nutzbar zu machen. Bei näherer Betrachtung stellt sich allerdings heraus, daß einige Architekturprinzipien (z.B. Master/Slave-Rolle), aber auch gegenwärtige Kartenbetriebssysteme, Defizite aufweisen, die die Integration erschweren. Im folgenden werden die diesbezüglich gewonnen Erfahrungen aus den realisierten Beispielszenarien geschildert und die für eine einfachere Anwendungsentwicklung nötigen Infrastrukturmaßnahmen vorgestellt und diskutiert.

5.1 Kartenlokale Infrastruktur

Idealerweise wird eine Chipkarte als ein eigenständiger Rechner im Netzverbund angesehen. Damit eine Karte so behandelt werden kann, muß sie jedoch eine ähnliche lokale Infrastruktur aufweisen wie ein herkömmlicher Rechner. Neben der rein physikalischen Ausstattung mit CPU und Speicher muß insbesondere das Betriebssystem auf die im Vergleich zur bisherigen Nutzung neuen Aufgaben eingerichtet sein. Hier bieten auch die neueren Kartenbetriebsysteme kaum adäquate Unterstützung: Es ist zwar möglich, ausführbaren Code fast jederzeit auf die Karte zu laden, bisher kann aber lediglich genau eine der auf der Karte befindlichen Applikationen aktiv sein. Auch gibt es prinzipiell die Möglichkeit, beim Start der Karte die Aktivierung einer bestimmten Applikation zu fordern, dieses Protokoll wird jedoch in der Praxis bisher kaum von den Kartenterminals benutzt.

Neben diesen eher organisatorischen Mängeln hat sich im Rahmen der prototypischen Realisierungen der Beispielszenarien gezeigt, daß die zur Zeit erhältlichen Betriebssysteme einen gravierenden Nachteil besitzen: es ist nicht vorgesehen, daß die Karte von sich aus aktiv wird. Tatsächlich werden Chipkarten bisher lediglich als „Slave“ zur Bereitstellung einer bestimmten Funktionalität angesehen: Hat die Karte auf eine Anforderung geantwortet, gilt der Dienst als erbracht und eine darüber hinausgehende Aktivität der Karte ist nicht vorgesehen. Um die Master/Slave-Rolle einfach vertauschen zu können und um echtes Multiprogramming zu erreichen, müßte auf den Karten ein für klassische Rechner übliches CPU-Scheduling, Interruptsteuerung, Speicherverwaltung, Speicherschutz sowie ein Dateisystem mit Zugriffsrechten realisiert werden. Der Standard ISO/IEC 7816 müßte dazu ergänzt bzw. geändert werden.

Ein weiteres Problem stellt der beschränkte Kommunikationsmechanismus zur Karte dar: Ergebnisdaten können aufgrund ihrer Größe u.U. nicht in einer einzigen Antwortnachricht übertragen werden und das ISO/IEC 7816-3 Protokoll sieht kein Chaining vor. Dieses Problem kann wiederum nur durch ein dem SIM-Toolkit-ähnlichen Konzept umgangen werden, indem das Ergebnis in mehreren Schritten paketisiert versandt wird. Dies ist umständlich und bei netzbasierten Applikationen bei einer Übertragungsrate von 9600 Baud zur Karte u.U. auch sehr zeitaufwendig.

Da Chipkarten aufgrund ihrer Beschaffenheit vielfach in sicherheitskritischen Bereichen eingesetzt werden, sind die meisten Chipkarten mit Krypto-Coprozessoren ausgestattet, die die in diesem Bereich anfallenden kryptographischen Berechnungen effizient ausführen. Im Rahmen einer Nutzung als Anwendungsplattform muß die Funktionalität des Coprozessors in geeigneter Form, zum Beispiel als Programmbibliothek, angeboten werden. Diese Unterstützung wird von den Karten zur Zeit nicht oder nur in eingeschränkter Form angeboten. In zukünftigen Szenarien könnten die Karten auch ihre spezifischen kryptographischen Fähigkeiten netzweit exportieren und zur Verfügung stellen - was u.U. aber politisch-rechtliche Probleme induziert.

Zusammenfassend läßt sich feststellen, daß die aktuellen Chipkarten mit ihren Betriebssystemen noch nicht die Fähigkeit einer universellen Nutzung in einem Netz anbieten. Damit die Vision einer effizienten und netzglobalen Verwendung Wirklichkeit werden kann, müssen insbesondere die Kartenbetriebssysteme noch eine ähnliche Entwicklung durchlaufen wie die Betriebssysteme herkömmlicher Rechner.

5.2 Infrastruktur außerhalb der Karte

Um eine Verfügbarkeit von Chipkarten als allgemeine Programmierplattform in einem offenen Netzverbund zu erreichen, sind auch Infrastrukturmaßnahmen im Umfeld der Karte erforderlich.

Auf die Chipkarte kann nur mittels Kartenleser zugegriffen werden. Damit muß jegliche Netzkommunikation mit der Karte durch den Kartenrechner vermittelt werden, der daher zusätzliche Vermittlungssoftware benötig. Damit der Zugriff und die Kommunikation mit verschiedenen Karten- und Kartenlesertypen in uniformer - und im Idealfall in transparenter - Weise geschehen kann, muß hier eine entsprechende Schnittstelle zur Verfügung gestellt werden, die einerseits die Ressourcen der Karte, und andererseits die Applikationen auf der Karte netzweit anbietet. Diese Anbindung kann, wie bisher meist üblich, applikationsspezifisch geschehen. In diesem Fall existiert zu jeder Kartenapplikation ein entsprechender Kommunikationspartner auf dem Kartenrechner, doch sollte dieser zweckmäßigerweise applikationsunabhängig sein und generischen Charakter besitzen.

Da in Form von Middleware (CORBA, JINI, DCOM etc.) bereits ähnliche, generische Schnittstellen und Mechanismen für klassische Rechnerverbundsysteme existieren, liegt es nahe, diese Mechanismen zu nutzen und die Kartenanbindung dort zu verankern. Wie die beschriebene prototypische Anbindung der Karte an die CORBA-Middleware zeigt, ist dies zur Zeit nur mit Einschränkungen möglich: Hier muß noch eine Vermittlungsinstanz auf dem Terminal eingesetzt werden. Es zeichnet sich allerdings ab, daß die Leistungsfähigkeit der auf Chipkarten verfügbaren Ressourcen ähnlich schnell zunehmen wird, wie dies im Bereich herkömmlicher Rechnertechnologie geschieht, so daß die Anbindung in absehbarer Zeit auch direkt auf der Karte gesche-

hen kann und das Terminal lediglich noch den physikalischen Netzanschluß für die Karte bereitstellen muß.

Aufgrund der zu erwartenden massenhaften Verbreitung und damit sehr großen Zahl von Chipkarten sind die heutigen Verfahren der Netzkomponentenadressierung dafür allerdings nicht mehr geeignet, da sie keinen ausreichend großen Adreßraum anbieten. Abhilfe kann hier der kommende Internetstandard IPv6 bieten, der mit seinem wesentlich vergrößerten Adreßraum die Möglichkeit eröffnet, jede Chipkarte mit einer eigenen IP-Adresse zu versehen.

6 Fazit

Leicht zu programmierende, multifunktionale Chipkarten stellen für zukünftige verteilte Applikationsszenarien ein interessantes Potential als Teil einer Anwendungsplattform dar. Durch die ihnen eigene Charakteristik der einfachen Anwendbarkeit und leichten Verbreitung bei gleichzeitiger Zusicherung verschiedener Sicherheitseigenschaften sind sie nicht nur als persönlicher und vertrauenswürdiger Kleinstrechner einsetzbar, sondern können auch als sichere Ausführungsplattform für Softwarekomponenten in Netzen dienen. Zwar ist sowohl die Hard- als auch die Software von Chipkarten für die Nutzung als Ausführungsplattform für verteilte Anwendungen noch nicht ideal, doch zeichnet sich aber ab, daß die weitere Entwicklung sehr schnell verlaufen könnte. Durch den geplanten Einsatz von multifunktionalen Chipkarten im Mobiltelefonbereich, der auch in Deutschland zur Zeit ebenso wie das Internet eine rasante Entwicklung durchläuft, wird die schnelle Verbreitung solcher Karten noch weiter gefördert. Es bleibt allerdings abzuwarten, wie zügig die hier beschriebenen Defizite abgebaut werden können und welche konkreten Applikationen diese neue Plattform in der Praxis hervorbringen wird.

Literatur

[1] W. Rankl, W. Effing, Handbuch der Chipkarten, Carl Hanser Verlag, 1999

[2] http://www.bluetooth.com

[3] http://java.sun.com/products/javacard

[4] http://www.javacardforum.org

[5] http://java.sun.com/products/jini/

[6] S. Fünfrocken, F. Mattern, Mobile Agents as an Architectural Concept for Internet-based Distributed Applications - The WASP Project Approach, in KiVS'99 (Hg.: Steinmetz), Springer-Verlag, pp. 32-43, 1999

[7] T. Lindholm, F. Yellin, The Java Virtual Machine Specification, Addison-Wesley, 1997

[8] J. Posegga, Java Smart Cards as a Platform for Electronic Commerce, in: Electronic Commerce - IFIP/GI Working Conference on Trends in Distributed Systems for Electronic Commerce (Hg.: Griffel, Tu, Lamersdorf), dpunkt-Verlag, pp. 175-182, 1998

[9] J. Posegga, H. Vogt, Byte Code Verification for Java Smart Cards Based on Model Checking, in: Proc. 5th European Symposium on Research in Computer Science - ESORICS 98 (Hg.: Quisquater, Deswarte, Meadows, Gollmann), Springer-Verlag, LNCS 1485, 1998

[10] GTS GSM 11.11 und GSM 11.14, Digital Cellular Telecommunication Systems (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface; GSM Technical Specification; March 1996