Java auf Chipkarten[1]

Matthias Kaiserswerth IBM Zürich Research Laboratory, Säumerstr.4
CH-8803 Rüschlikon, kai@zurich.ibm.com

Joachim Posegga Deutsche Telekom AG, Technologiezentum,
D-64276 Darmstadt, posegga@tzd.telekom.de

Java-Technologie dringt derzeit in immer mehr Bereiche vor: Java-fähige Web-Browser sind inzwischen eine Selbstverständlichkeit geworden, und auch die Aussicht, daß 1998 die ersten Telefone mit Java-Unterstützung auf dem Markt kommen werden, erregt heute kaum noch Aufsehen.

Selbst für manche Java-Insider überraschend ist jedoch die Entwicklung im Bereich Java-fähiger Chipkarten: Ende Oktober 1996 wurde von JavaSoft (mit technischen Beiträgen von Schlumberger) das sogenannte JavaCard API V1.0 publiziert und im Mai 1997 war der erste Prototyp einer Java-fähigen Chipkarte verfügbar (siehe http://www.cyberflex.austin.et.slb.com/). In der Zwischenzeit arbeitet JavaSoft zusammen mit verschiedenen Technologiepartnern unter anderem dem JavaCard Forum (http://www.javacardforum.org) an einer Version 2 dieser Spezifikation.

Es stellt sich die Frage, was hinter dieser Technologie steckt, und was von ihr zu erwarten ist.

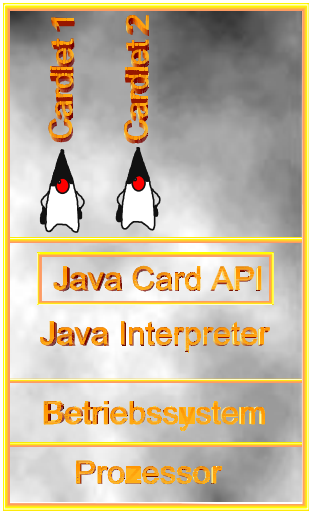
Aufbau einer Java-Chipkarte

Zunächst ist

Die dafür notwendige Java-Laufzeitplattform („Virtual Machine“ VM) läßt sich in einer Chipkarte entweder in Form von Software oder als Hardware realisieren. Letzteres z.B. durch Integration des Chip-Layouts „Pico-Java“ von Sun Microsystems. Eine solche Hardwareimplementierung verspricht gegenüber einer Softwareemulation eine höhere Verarbeitungsgeschwindigkeit, die Realisierung ist aber aufwendiger und dürfte frühestens Ende 1998 zu erwarten sein.

[1] Wir danken Herrn Dr. Frank Schäfer-Lorinser (Deutsche Telekom, TZ Darmstadt) für nützliche Anregungen.

Die zur Zeit verfügbare JavaCard [1] „Cyberflex“ der Firma Schlumberger ist eine Softwarelösung. Der prinzipielle Aufbau einer solchen Karte ist in nebenstehender Abbildung dargestellt: Grundlage ist ein handelsüblicher 8-bit Chipkartenprozessor (basierend auf Intel 8051 oder Motorola 6805) mit einem Betriebssystem nach ISO-7816. Darauf ist eine JavaCard VM implementiert, die die Java-Anwendungen (sog. „Cardlets“) interpretiert, und das JavaCard API, u.a. für den Zugriff auf die in ISO 7816-4 spezifizierten Dienste, zur Verfügung stellt.



Der Speicherplatz in einer Chipkarte ist sehr beschränkt: im Falle von Cyberflex benötigt das Betriebssystem ca. 8 Kb, der Java-Interpreter 4 Kb, das API 1.6 Kb. Bei der heutigen Technologie stehen dann noch etwa 2.8 Kb im EEPROM für Cardlets zur Verfügung, die sich mit ca. 200 Byte RAM begnügen müssen.

Die derzeitige sowie auch die geplante Version 2 der JavaCard Spezifikation unterstützt nicht den vollen Sprachumfang von Java, sondern stellt lediglich eine Untermenge zur Verfügung. Es gibt u.a. keine 32-bit Ganzzahlarithmetik, keine Gleitkommaarithmetik, keine dynamische Speicherverwaltung, keine Ausnahmebehandlung und keine nebenläufigen Routinen (Threads).

Wozu Java in einer Chipkarte?

Die Möglichkeit, Java-Programme in einer Chipkarte ablaufen zu lassen, ist sicher spektakulär, aber welche tatsächlichen Vorteile bietet ein solches Szenario? Diese Frage stellt sich insbesondere, da Java hier nicht als „downloadable content“ (Java-Applets) eingesetzt wird, sondern lediglich als Programmiersprache für die Chipkarte dient.

Eine der wichtigsten Kundenforderungen ist die Möglichkeit, Anwendungen auf einer Chipkarte auch nach ihrer Initialisierung bzw. Personalisierung leicht austauschen oder neu installieren zu können. Dies ist aus zwei Gesichtspunkten heraus interessant: Zum einen vereinfacht es die Realisierung von Kartenanwendungen erheblich, da das Einbringen der Software in die Karte nicht mehr mit dem Kartenbetriebssystemhersteller beim Erzeugen der ROM-Maske koordiniert werden muß. Dies war bisher oft notwendig, da aus Platzmangel das Betriebssystem der Karte auf die Anwendung zugeschnitten werden mußte. Java verspricht nun, durch eine klare Trennung zwischen Anwendungs- und Betriebssystemprogrammierung, wesentlich schneller und flexibler am Markt reagieren zu können. Eine weitere Anforderung ist es, Updates der Software auf ausgegebenen Karten durchzuführen, oder auch eine Karte an neue Anwendungsbereiche anzupassen. Mit der derzeit gängigen Technologie ist die Veränderung der Kartensoftware bei einmal ausgegebenen Karten nur noch sehr schwer möglich und setzt genaue Kenntnisse des Kartenbetriebssytems voraus.

Die Verwendung von Java weist, daher gegenüber der gängigen Technologie zwei entscheidende Vorteile auf:

1. *Plattformunabhängigkeit: Cardlets sind nicht an bestimmte Chipkarten gebunden, sondern können in jeder Java-fähigen Chipkarte (unabhängig vom Kartenbetriebssystemhersteller und dem zugrundeliegenden Prozessor) genutzt werden.*

2. *Im Vergleich zur herkömmlichen Chipkartenprogrammierung bietet Java Garantien für erhöhte Sicherheit wie im folgenden Abschnitt kurz beschrieben werden soll.*

Sicherheitsaspekte

Sicherheit spielt bei Chipkarten eine wesentliche Rolle, denn eine Chipkarte enthält oft hochsensible Informationen (z.B. geheime Schlüssel), die nicht nach außen gelangen dürfen oder eine elektronische Geldbörse, bei der sich der Inhalt nicht auf unkontrollierte Weise verändern sollte. Wird nun bei einer Java-Chipkarte eine neue Anwendung in das Innere der Karte geladen, so muß sichergestellt werden, daß diese Anwendung nur mit ihren eigenen Daten arbeitet und Daten bzw. Programmkode anderer Anwendungen weder lesen, verändern noch löschen kann.

Das Sicherheitsmodell von Java spielt dabei eine wesentliche Rolle, denn es verhindert den direkten Zugriff auf Information außerhalb der durch die virtuelle Maschine definierten Grenzen (Sandbox). Dies kann jedoch nur dann garantiert werden, wenn die Implementierung der Java-VM fehlerfrei ist. Arbeiten, die in diese Richtung zielen, stehen jedoch erst in den Anfängen (siehe [2], http://www.cli.com/software/djvm/).

Wenn unterschiedliche Anwendungen miteinander Daten austauschen sollen, so soll dies durch das JavaCard API in kontrollierter Weise gewährleistet werden. Dabei stellt man sich bei Karten, die z.B. im Finanzbereich eingesetzt werden sollen, vor, daß die Cardlets mit einer digitalen Signatur versehen werden. Bei der Instantiierung auf der Karte könnte dann zum einen die Integrität der Anwendung überprüft werden und zum anderen ließen sich aus der Unterschrift gewisse Zugriffsrechte für das Cardlet ableiten. Über ähnliche Mechanismen ("Secure Messaging") ließe sich auch das eigentliche Laden von Cardlets nur auf den Kartenherausgeber beschränken.

Literatur

[1] Scott Guthry: *Java Card: Internet Computing on a Smart Card.* IEEE Internet Computing, Vol. 1, No. 1, pp. 57-59, 1997.

[2] Richard M. Cohen: *The defensive Java Virtual Machne Specification,* Technical report, draft Rrlease, Computational Logic Inc., Austin, TX, 1997. http://www.cli.com/software/djvm/html-0.5/djvm-report.html

[3] Sun Microsystems: *Java Card 2.0 Language Subset and Virtual Machine Specification*. 1997. http://java.sun.com/products/javacard/index.html

[4] Sun Microsystems: *Java Card 2.0 Application Programming Internfaces.* 1997. http://java.sun.com/products/javacard/index.html