

Praxiseinsatz von Java in eingebetteten Systemen

Java auf Systemen mit eingeschränkten Ressourcen – Chancen und Herausforderungen

Lange war die Entwicklung von Softwarelösungen für Embedded-Systeme eine Domäne, in der hardwarenahes und hochspezialisiertes Fachwissen erforderlich waren. Durch den Einsatz von Java-Technologie ist es unterdessen auch in diesem Umfeld für Softwareingenieure mit breiterem Hintergrund möglich, leistungsfähige Anwendungen zu realisieren. Gleichzeitig bietet dieser Ansatz eine grosse Fülle an modernen Möglichkeiten, die erst durch die neue Technologie in der Praxis möglich werden. Der vorliegende Beitrag geht auf diese Chancen, aber auch auf die damit verbundenen Herausforderungen ein, gibt aktuelle Beispiele aus dem Markt und liefert Praxiserfahrungen, die Ergon Informatik in Entwicklungsprojekten mit Embedded Java gemacht hat.

In der vergangenen Zeit hat der Trend, Java-Technologie auch in Embedded Devices einzusetzen, stark an Bedeutung gewonnen. Einige Ankündigungen und Ver-

Peter K. Brandt

öffentlichungen von Sun Microsystems, aber auch Hardware-Neuentwicklungen von europäischen Firmen zeigen deutlich, dass Java heute auch für den Einsatz auf sehr kleinen Geräten bereit ist, die vom Endanwender häufig nicht als eigentliche Computer wahrgenommen werden (Bild 1). Wie in diesem Artikel gezeigt wird, eröffnet diese Tatsache eine grosse Anzahl neuer Möglichkeiten für den Entwurf, die Umsetzung, den Einsatz und die Evolution von Embedded Systems.

Bisherige Technologie

Embedded Systems, also Computersysteme, die auf einen vorgegebenen Einsatzzweck massgeschneidert sind, findet man heute an vielen Orten. Von der intelligenten Waschmaschine über Klimasteuerungen, Autos, Kaffeemaschinen und Roboter bis hin zu Sensornetzwerken: Alle enthalten kleine Computer, die in das eigentliche Gerät eingebettet sind und seine Funktionsweise steuern. Typischerweise handelt es sich um ein System, das entworfen ist, um möglichst autonom und aus-

fallfrei seine Aufgaben zu erledigen. Oft sind auch zeitkritische Aufgaben Teil des Anforderungskatalogs.

Mit herkömmlichen Technologien wurde die Entwicklung von Embedded Devices bislang sehr maschinennah vorgenommen: Unter Anwendung spezialisierter Werkzeuge und spezifischer Entwicklungsumge-

bungen wird in Sprachen wie C, C++ oder gar Assembler eine für den Einsatzzweck des Geräts massgeschneiderte Anwendung erstellt. Dieses Vorgehen erfordert vom Entwickler einerseits detaillierte Kenntnisse über die Programmierung in eingeschränkten Umgebungen, aber auch ganz spezifische Fähigkeiten im Umgang mit dem eingesetzten Betriebssystem, der verwendeten Hardware (CPU, Speichermanagement, etc.) und den zu benutzenden Entwicklungstools. Im Gegenzug ist es andererseits auf diese Weise möglich, das resultierende System und seine Ressourcennutzung bis ins letzte Detail zu optimieren. Ob dies immer nötig und sinnvoll ist, hängt sicher vom konkreten Einsatzbereich ab.

Die oben genannten Einschränkungen und Voraussetzungen bei der Entwicklung stellen eine grosse Einstiegshürde in diesem Bereich dar und führen dazu, dass sich die Entwickler von Embedded Systems auf Basis traditioneller Technologien ein ausgeprägtes Spezialwissen aneignen müssen. Dies selbst im häufigen Fall, in dem aufgrund unkritischer Projektanforderungen hardwarenahe Detailkenntnisse gar nicht erforderlich wären.



Bild 1 SunSpot-Devices: Spontan drahtlos vernetzte und mobile Java-Kleinstcomputer auf hohem Leistungsniveau.

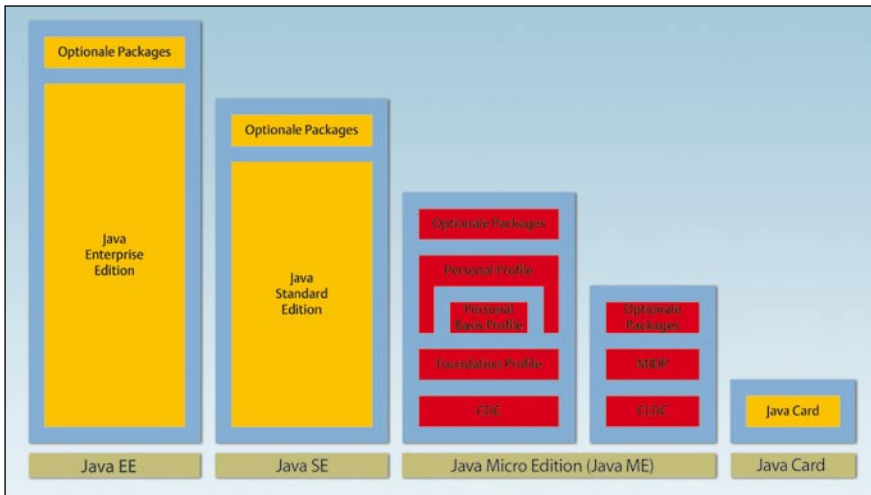


Bild 2 Die Java-Familie – massgeschneiderte Varianten für jede Umgebung.

So resultieren schliesslich beim Einsatz herkömmlicher Methoden eingebettete Systeme, die zwar stark auf einen spezifischen Anwendungsfall hin optimiert sind, gleichzeitig jedoch abgeschlossene Komponenten darstellen. Eine dynamische Erweiterung solcher Systeme um neue Funktionalität durch Systemintegratoren oder gar IT-Spezialisten der Endanwender ist daher kaum anzutreffen. Genau dies könnte jedoch dem Anwender einen wesentlichen Nutzen bringen – und die Marktchancen eines Produkts deutlich erhöhen.

Java massgeschneidert – von den Ursprüngen bis heute

Java entstand Anfang der 1990er-Jahre im Rahmen des Green-Projekts bei Sun Microsystems, das «the next wave in computing» erkennen und vorbereiten sollte [1]. Zur Zeit der ersten öffentlichen Vorstellung stand der Fokus vor allem auf interaktiven Internetinhalten. Doch innert kurzer Zeit war klar, dass Java weit mehr Potenzial besass als zur Animation von Webseiten.

Aus der einfachen, aber leistungsfähigen Programmiersprache entstand eine umfassende Technologiefamilie, die das volle Spektrum von Desktopapplikationen zu Serveranwendungen bis hin zu Software auf Mobiltelefonen und Chipkarten abzudecken vermag (Bild 2). Da die verschiedenen Gebiete stark unterschiedliche Anforderungen haben, wurde eine Reihe von Standards geschaffen, die auf den jeweiligen Einsatzbereich massgeschneidert und gleichzeitig interoperabel mit den anderen Gebieten sind: Java SE, die Standard Edition, fokussiert sich auf Desktopanwendungen; Java EE, die Enterprise Edition, definiert leistungsfähige Technologien zum Einsatz im Serverumfeld; Java ME, die Micro Edition, beschreibt Varianten der

Java-Technologie für ressourceneingeschränkte Umgebungen; und Java Card schliesslich spezifiziert eine Java-kompatible Umgebung für intelligente Chipkarten.

Java ME, die Java Micro Edition, wurde lange als Technologie für portable Applikationen für Mobiltelefone wahrgenommen. Doch die Ausrichtung liegt grundsätzlich bei allen Laufzeitumgebungen, für die die Standard Edition zu grosse Herausforderungen stellen würde. Angesprochen sind also neben Mobiltelefonen auch PDAs, Set-topboxen und beliebige Arten von eingebetteten Systemen, sei es mit oder auch ohne Benutzerschnittstelle.

Da sich die Zielplattformen gerade im Bereich der Micro Edition stark voneinander unterscheiden, wurde eine Unterteilung in sogenannte Configurations vorgenommen. Die Connected Device Configuration (CDC) definiert eine Klasse von leistungsfähigeren Endgeräten, beispielsweise PDAs oder Highend-Smartphones. Die Connected Limited Device Configuration (CLDC) hingegen zielt ab auf noch eingeschränktere Endgeräte. Auf beide Configurations baut eine Reihe von Profiles auf, die zusätzliche Features definieren, beispielsweise das Personal Profile (PP) auf CDC oder das Mobile Information Device Profile (MIDP), das in Mobiltelefonen über CLDC zum Ein-

Webtechnologien auch für Embedded Systems

Was im Desktop- und Serverumfeld seit Langem üblich ist, kann durch die Java-Technologien auf einfache Weise nun auch in eingebetteten Systemen gängige Praxis werden: der Einsatz von Komponentenarchitekturen und Webtechnologien. Die Java-Plattform definiert als wesentlichen technologischen Bestandteil eine virtuelle Maschine, die sogenannte Java Virtual Machine (JVM). Diese JVM ist ein virtueller Rechner für die Ausführung von Programmen, der bereits als grundlegende Eigenschaft objektorientierte Programmierung und dynamisches Laden von Programmkomponenten unterstützt.

Diese Konzepte können nun verwendet werden, um leistungsfähige, zur Laufzeit erweiterbare Frameworks und Komponentenarchitekturen zu definieren. Diese halten zurzeit Einzug auf kleinen und kleinsten Systemen, also im Umfeld der Java Micro Edition, die auch für Embedded Systems eingesetzt wird.

Ein prominentes Beispiel ist die OSGi-Service-Plattform [2], die es erlaubt, zur Laufzeit dynamisch und kontrolliert Serviceanwendungen, sogenannte «Bundles», einzuspielen, zu aktualisieren und zu entfer-

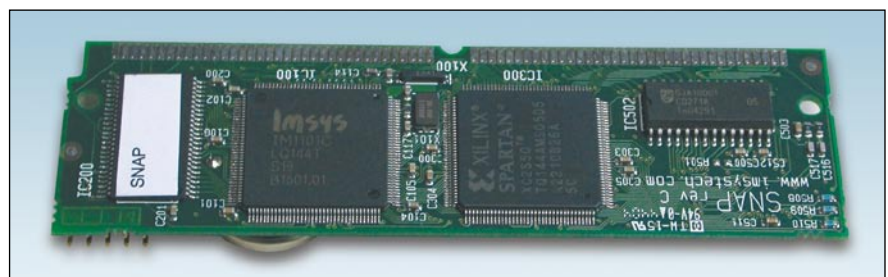


Bild 3 Imsys-Java-Microcontroller auf SNAP-Referenzplattform. Der Chip wurde unterdessen weiter verkleinert.

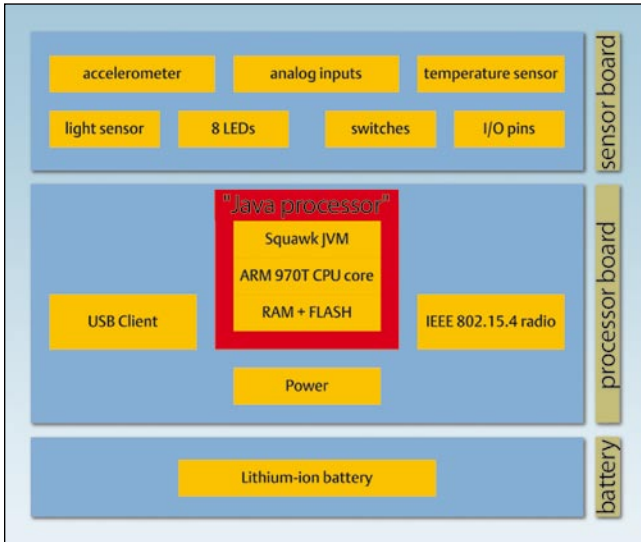


Bild 4 Aufbau der SunSpot-Devices.

Frameworkarchitekturen, dynamischen Plug-in-Mechanismen und hochgradig vernetzten Services ist sicher der Produktivitätsgewinn nicht zu vernachlässigen, der dadurch entsteht, dass auf allen Systemebenen die gleiche Technologie zum Einsatz kommt. Dies resultiert nicht nur in einer besseren Wiederverwendbarkeit von Programmcode, sondern auch in geringerem Aufwand durch einheitliche und bereits bekannte Entwicklungstools. Zuletzt ist sicher auch die Verfügbarkeit einer grossen Zahl von gut ausgebildeten und motivierten Java-Entwicklern ein Aspekt, der für den Einsatz von Java-Technologie spricht, statt hardwarenaher Realisierung in einem Embedded-Projekt.

nen, auch aus der Ferne. Grosses Augenmerk wurde dabei einem robusten Versionsierungsmechanismus geschenkt, um die aus anderen Systemen bekannte und benötigte «DLL-hell» zu vermeiden.

Vorteile von Java

Java als Laufzeitumgebung von Embedded Systems liefert gleichzeitig eine solide Basis für verteilte Systeme, da die gängigen Internettechnologien quasi von Haus aus unterstützt werden. Ein prominentes Beispiel dafür ist der Servlet-Container, der Bestandteil eines gängigen Java-Embedded-Produkts ist. Auf dieser Basis können Standard-Webapplikationen nun auf einfache Weise Teil eines Embedded System werden. Gleichzeitig ist es möglich, die Menge der deployten Webapplikationen mit gängigen Mechanismen zu erweitern, bei Bedarf selbst zur Laufzeit eines Embedded System im Feld.

Dies wiederum öffnet Tür und Tor für eine breite Palette von vernetzten und kollaborativen Services, die von vielen Embedded Devices gemeinsam erbracht werden und die dem Benutzer einen Gesamtnutzen bringen, der über die Summe der Einzel-fähigkeiten hinausgeht. So können beispielsweise auf einfache Weise verteilte Regelungsanwendungen erstellt werden, deren Verhalten von einer Vielzahl von Netzwerkknoten mit beeinflusst wird.

Auch für das häufig benötigte Monitoring- und Konfigurationsinterface im Web-Browser ist auf diesem Weg die Grundlage vorhanden. In Kombination mit anderen heute gängigen Technologien wie Ajax (siehe unten) werden Benutzerschnittstellen möglich, die trotz geringer Leistung der involvierten Embedded Devices eine enorme Performance und Bedienerfreundlichkeit aufweisen.

Neben den oben genannten Aspekten der Unterstützung von leistungsfähigen

Risikofaktoren

Bei allen Vorteilen, die die beschriebene Technologie in der Realisierung von eingebetteten Systemen verspricht, dürfen die Risiken nicht ausser Acht gelassen werden. Aufgrund der dynamischen Natur der Java-Laufzeitumgebung ist die Möglichkeit, exakte Vorhersagen über das Laufzeitverhalten der erstellten Software zu machen, in der Regel eingeschränkt. Insbesondere falls Echtzeitanforderungen wichtiger Bestandteil der System-Requirements sind, können diese in der Praxis eine Restriktion darstellen, die anderen Technologien einen klaren Vorteil gibt.

Auch Anforderungen wie extrem hohe Performance bei gleichzeitig starkem Kostendruck können für den Einsatz von traditionellen Realisierungsvarianten sprechen. Daher sollten vor Einsätzen der Java-Technologie unter solchen Voraussetzungen seriöse Machbarkeitsabklärungen stehen, um sicherzustellen, dass die vorgesehene Technologie der Problemstellung gewachsen ist.

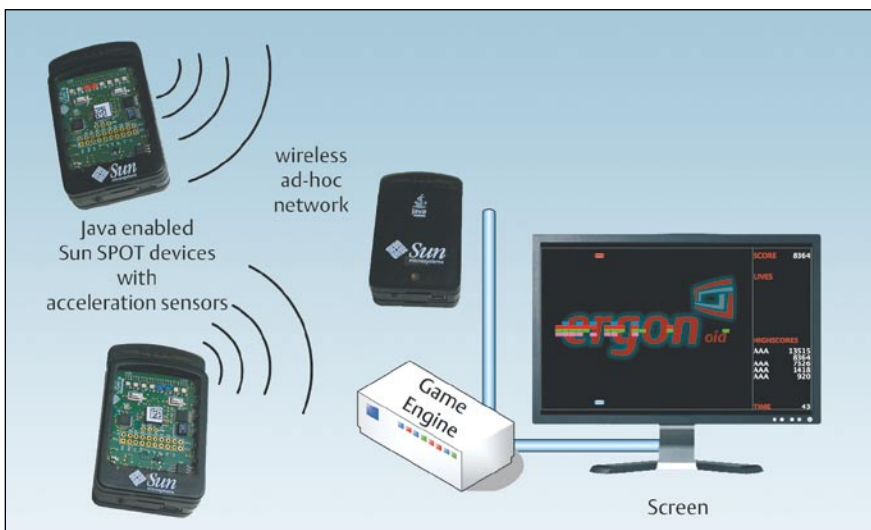


Bild 5 Zusammenspiel der Devices im Ergonoid-System.

Typische kommerziell erhältliche Embedded Java Devices

Unterdessen sind auf dem Markt verschiedene Java-Komponenten für Embedded Devices verfügbar. Im Folgenden sollen zwei Beispiele vorgestellt werden, mit denen wir in den letzten Jahren Erfahrungen sammeln konnten, der Java-Mikroprozessor IM1101 der schwedischen Imsys Technologies [3] sowie die SunSpot Embedded Devices von Sun Microsystems [4].

Der IM1101/SNAP führt Java-Applikationen (Java ME, CLDC) native aus und beinhaltet neben einer grossen Zahl von externen Interfaces ein einfaches Betriebssystem, einen TCP/IP-Stack sowie einen in Java implementierten Servlet-Container/ Webserver. Letzterer ermöglicht es auf ein-

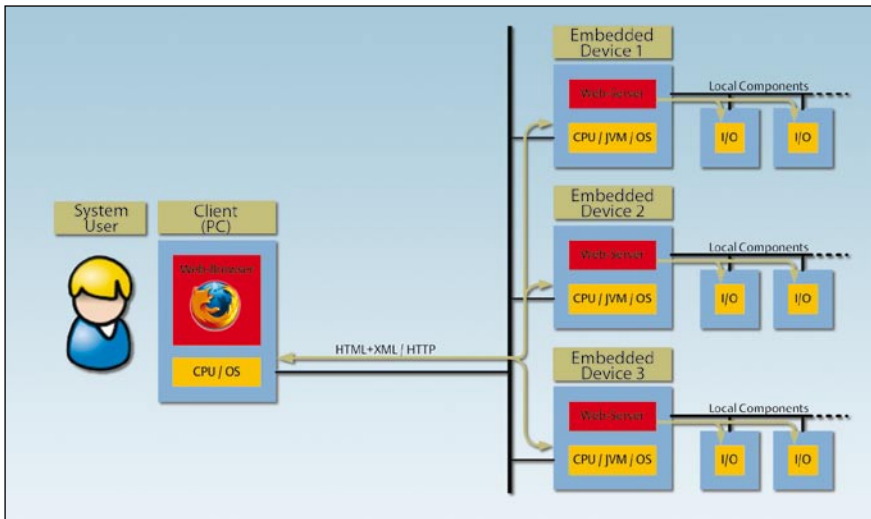


Bild 6 Direktzugriff auf Devices im Feld vom Client PC aus.

fache Weise, selbst sehr kleine Embedded Devices mit einem Webinterface basierend auf Standardtechnologien auszurüsten.

Neben der Möglichkeit, das Device mit Standard-Java-Tools zu programmieren, lässt sich der Mikrocode für spezialisierte Anwendungen und um maximale Performance zu erreichen, auch um eigene Funktionen erweitern. Zum raschen Einstieg wird vom Hersteller ein Entwicklungsboard, die SNAP-Referenzplattform, angeboten, das einfachen Zugriff auf die Peripherie ermöglicht und stand alone eingesetzt werden kann. Bei Ergon ist der Mikrocontroller für die Erstellung einer verteilten Regelungsanwendung für einen Kunden aus dem industriellen Umfeld im Einsatz (Bild 3).

Die SunSpot-Devices von Sun Microsystems beinhalten ebenfalls eine Reihe von Sensoren und Ein-/Ausgabekanälen wie Temperatursensor, Helligkeitssensor, Beschleunigungssensor sowie analogen

und digitalen Ein- und Ausgängen. Die Komponenten sind modular aufgebaut aus einem Processor-Board, das die eigentliche CPU, ein USB-Interface sowie ein IEEE-802.15.4-kompatibles Funkmodul enthält (Zigbee), sowie optionalen Erweiterungsboards, von denen das oben genannte Sensorboard ein Beispiel ist. Über ein standardisiertes Interface lassen sich andere, selbst entwickelte Peripherieboards anschließen. Der grundsätzliche Aufbau des Device ist in Bild 4 dargestellt.

Auf der Basis dieser Devices wurde von uns anlässlich der Jazoon-Konferenz 2007 eine Anwendung entwickelt, die verschiedene Fähigkeiten der Hardware nutzt. Mittels der Beschleunigungssensoren wird die Lage der Komponenten im Raum ermittelt und via Funkschnittstelle an eine (ebenfalls auf SunSpot laufende) Basisstation übertragen. Diese steuert mit der empfangenen Information die grafische Darstellung von

Ergonoid, einem Klon eines Spielhallenklassikers (Bild 5). Zudem wird über die Funkschnittstelle und die integrierten LED-Zeilen Feedback an die Benutzer zurückgegeben. Bei diesem Einsatz wurde aufgrund der hervorragenden Benutzerinteraktion deutlich, zu welcher Leistung heutige Java-basierte Embedded Devices in der Lage sind.

Projekteinsatz bei Ergon

Neben diesem eher prototypischen Einsatz wird bei uns, wie oben erwähnt, seit längerem Embedded-Java-Technologie im Projektumfeld eingesetzt. Dort wird mit verschiedenen Techniken die Interaktion einer grossen Zahl von Embedded Devices nutzbringend angewendet. Java hat sich hier bewährt. Durch den Einsatz gleicher Technologien auf Server, Embedded Device und anderen Systemkomponenten wird eine deutliche Steigerung der Flexibilität erreicht. So ist es beispielsweise möglich, ein bestimmtes Protokoll, mit dem auf die Devices zur Konfiguration und Überwachung zugegriffen werden soll, direkt auf den Embedded Devices zu realisieren und von einem Client darauf zuzugreifen (Bild 6).

Andererseits ist es auch möglich, die Java-Implementierung in ähnlicher Form auf einem zentralisierten Server einzusetzen, der dann über ein devicespezifisches Protokoll auf die Endgeräte zugreift. Selbstverständlich sind gemischte Szenarien denkbar, bei denen der zentralisierte Server lediglich für simplere Komponenten die Proxy/Protokollkonverter-Funktion wahrnimmt und leistungsfähigere Embedded Devices das Protokoll selbst implementieren (Bild 7).

Die Variante ohne den zusätzlichen Server bringt durch den Wegfall der zentralen Komponente eine erhöhte Ausfallsicherheit des Gesamtsystems mit sich. Andererseits hat der Einsatz dieser Zwischenkomponente den Vorteil, dass an die beteiligten Devices weniger Anforderungen gestellt werden müssen und die Flexibilität für Änderungen am Rand des Systems wächst. Die im Bild gezeigte Anwendung Web-GUI zur Konfiguration durch einen Menschen ist hier nur ein Beispiel, stellvertretend für beliebige komplexere Protokolle. Ähnliche Konzepte können auch bei der Implementierung von Protokollen zur Machine-to-Machine-Interaktion eingesetzt werden.

Bei der Realisierung von browserbasierten Benutzerschnittstellen auf den im Feld verteilten und oft schwer zugänglichen Devices bewährt sich im vorliegenden Umfeld der Ajax-Ansatz. Da häufig die Rechenleistung des Client PC diejenige der Embedded Devices um ein Vielfaches übersteigt, macht es Sinn, viele Aufgaben der Web-

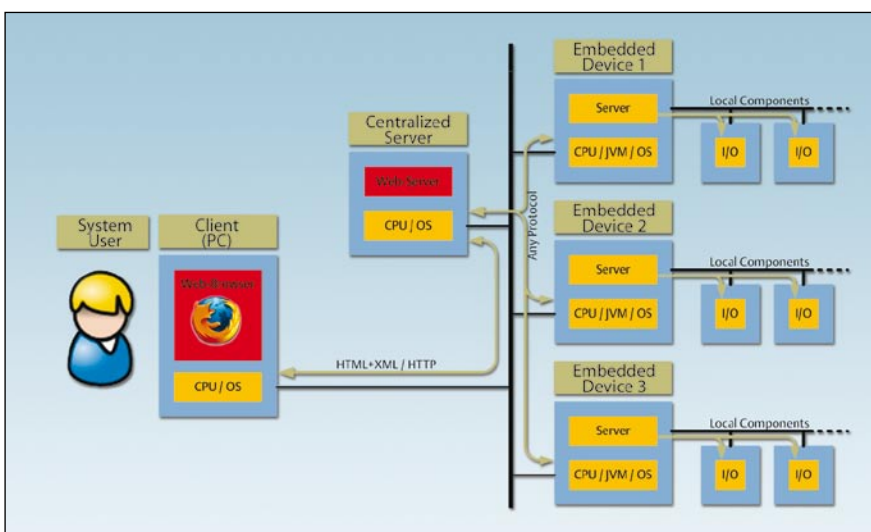


Bild 7 Zugriff auf Devices via Proxy-Server/Protokolladapter. Mischformen mit der Variante Direktzugriff sind möglich.

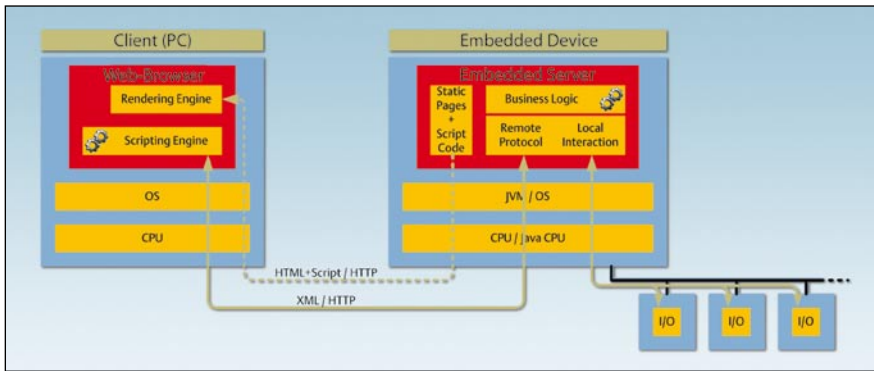


Bild 8 Aufgabenteilung zwischen starkem Desktop-Client und kleinem Server auf Embedded Device.

scheiden, und Schnittstellen zwischen den Systemteilen sind sehr natürlich.

Fazit

Zusammenfassend kann festgestellt werden, dass die von uns gemachten Erfahrungen grundsätzlich für den Einsatz von Java in Embedded Systems sprechen. Die Technologie hat einen Reifegrad erlangt, der die Verwendung in produktiven Systemen erlaubt. Ein wichtiger Schritt beim Systemdesign ist jedoch die Validierung, ob alle Voraussetzungen im konkreten Anwendungsfall gegeben sind. Der potenzielle Nutzen – gerade aus Sicht der Softwareentwicklung –, ein Embedded System auf Grundlage von Java-Technologie zu erstellen, ist hoch und kann ganz klar wettbewerbsentscheidend sein.

Referenzen

- [1] <http://java.sun.com/features/1998/05/birthday.html>
- [2] <http://www.osgi.org/>
- [3] <http://www.imsys.se/>
- [4] <http://www.sunspotworld.com/>

Angaben zum Autor

Peter K. Brandt ist Senior Software Engineer und Business Developer bei der Zürcher Firma Ergon Informatik AG. Nach dem Studium der Informatik an der ETH Zürich und der TU München war er als wissenschaftlicher Mitarbeiter am Institut für Computersysteme der ETH Zürich tätig. Im Anschluss daran befasste er sich während mehreren Jahren als Research Engineer bei Swisscom Innovations mit anwendungsbezogenen Forschungsthemen im Umfeld von Mobilität und Kommunikation. Seit 2001 ist Peter K. Brandt Mitarbeiter bei Ergon und dort für das Gebiet vernetzter mobiler Anwendungen auf Java-Basis verantwortlich. Ergon Informatik AG, 8008 Zürich, peter.brandt (at) ergon.ch

Die Ergon Informatik AG (www.ergon.ch) wurde 1984 gegründet und beschäftigt heute rund 100 Mitarbeiterinnen und Mitarbeiter. Das Unternehmen bezeichnet sich als führender unabhängiger Anbieter von Software-Engineering und ist spezialisiert auf Design, Entwicklung und Implementierung von hochstehenden massgeschneiderten Softwarelösungen auf der Basis offener Systeme. Als Spezialist im Bereich Java und als Pionier in der Entwicklung von Mobileapplikationen genießt Ergon laut eigenen Angaben weltweites Ansehen.

applikation auf den Client auszulagern. Auf diese Weise wird der Presentation Layer praktisch vollständig auf dem Client ausgeführt, während die zentralen Teile der Business-Logik auf dem Embedded Device verbleiben (Bild 8). Daher wird beim ersten Zugriff des PC-Clients lediglich eine simple HTML-Seite und eine Menge an Script-Code zum Browser übertragen, der anschliessend via XML-Requests mit den Embedded Devices kommuniziert und für die effiziente lokale GUI-Interaktion sorgt. Auf diese Weise hat die Servlet-Komponente auf dem Embedded Device lediglich die eigentlichen Nutzdaten in einem kompakten Format bereitzustellen und auf diese Weise zur Benutzerinteraktion beizutragen. Das Benutzererlebnis wird so stark verbessert.

Erfahrungen

Bei der Arbeit mit Java-basierten Embedded Systems in verschiedenen Projekten haben wir eine Reihe von Erfahrungen gesammelt. Das erste grosse Aha-Erlebnis tritt bei den beteiligten Entwicklern typischerweise nach kurzer Zeit mit der Arbeit mit Embedded Systems auf: Sie fühlen sich im neuen Umfeld sofort zu Hause und können mit ihren gewohnten Tools, den bekannten Konzepten und Methoden unmittelbar produktiv arbeiten. Beispielsweise ist es aus eigener Erfahrung auf Basis der oben erwähnten SunSpot-Devices möglich, ohne Vorkenntnisse im Embedded-Bereich innert weniger Stunden eine sinnvolle Applikation zu erstellen, die Temperatur-, Helligkeits- und Beschleunigungssensor sowie die eingebaute Funkschnittstelle nutzt und dabei sogar zur Erhöhung der Sendereichweite ein Proxying über andere Mobile/Embedded-Knoten realisiert.

Dieser leichte Einstieg ist sicherlich der Tatsache zu verdanken, dass sich die im Embedded-Umfeld häufig anzutreffende Variante Java ME sauber in die restliche Java-Welt einfügt. Dennoch – oder gerade deswegen – ist es essenziell, dass nicht

vergessen geht, dass es sich bei der Zielplattform um ein kleines Embedded Device handelt, das in der Regel wenig Leistung und Ressourcen anbietet. Selbst wenn beispielsweise eine Embedded-Plattform auf Java-ME-Grundlage einen Servlet Container anbietet, ist dieser von der Leistung her nicht mit einem Enterprise Servlet Container vergleichbar, auch wenn aus Implementierungssicht der Unterschied kaum in Erscheinung tritt. In dieser Hinsicht ist vorgängige Erfahrung im Umgang mit anderen Java-ME-Umgebungen (z.B. Applikationsentwicklung für Mobiltelefonanwendungen) sicher von Vorteil, wenn auch nicht zwingend erforderlich.

Gleichzeitig ist die Bandbreite in der Leistungsfähigkeit der Endgeräte gross und schwankt zwischen stark eingeschränkt und äusserst leistungsfähig. Daher ist es essenziell, dass vor einer Projektentscheidung für eine konkrete Plattform grundsätzliche Abklärungen betreffend benötigter und angebotener Performances stattfinden. Auch was die Integration der Systembestandteile auf den unterschiedlichen Komponenten (Server, Desktop, Mobile/Embedded) angeht, wird die einheitliche Technologie als sehr integrationsfördernd empfunden. Code-Komponenten können einfach auf anderen Knoten wieder verwendet werden, selbst wenn sich diese in der Leistung um Grössenordnungen unter-

Résumé

Utilisation pratique de Java dans les systèmes embarqués

Java sur les systèmes à ressources limitées – chances et défis. Le développement de solutions logicielles pour systèmes embarqués – Embedded Systems – a longtemps été un domaine exigeant des connaissances hautement spécialisées et touchant au domaine du matériel. Grâce à la technologie Java, les ingénieurs en logiciel disposant d'une expérience étendue peuvent maintenant réaliser des applications performantes. En même temps, cette approche offre une foule de possibilités modernes qui ne sont possibles en pratique que grâce à la nouvelle technologie. L'article étudie ces chances, de même que les risques correspondants, et présente des exemples du marché actuel et des expériences pratiques que la société Ergon Informatik a faites dans des projets de développement avec Embedded Java.