

aicas news

Neuigkeiten für Softwareentwickler kritischer Anwendungen

Frühling 2005

↓ Editorial

Lieber Leser,

ich freue mich, Ihnen die jüngsten Neuigkeiten und technischen Informationen von aicas zu präsentieren. Viel ist geschehen seit unserer letzten Ausgabe: die Bedienung und Funktionalität der JamaicaVM wurde stetig verbessert und für unsere amerikanischen Kunden bieten wir jetzt direkte Betreuung durch unsere US-Niederlassung. Aber keine aicas news wäre vollständig ohne einen tieferen Einblick in die Java-Echtzeittechnologie.

Unser Hauptartikel beleuchtet die Arbeiten, um Java-Technologie für den sicherheitskritischen Einsatz zu optimieren. Dies ist ein sorgfältiger Kompromiss zwischen dynamischem Verhalten und Sicherheit. Verbesserungen in diesem Bereich erweitern das Einsatzspektrum von Java.

Ich hoffe diese Ausgabe entspricht Ihren Erwartungen für einen informativen Artikel. Wir freuen uns auf Ihre Rückmeldungen und sind besonders an Ihren technischen Belangen interessiert.

Ihr Dr. James J. Hunt
CEO

JamaicaVM 2.6

Version 2.6 beinhaltet mehr Funktionen für eingebettete Systeme, ist weniger ressourcenintensiv und verbessert die Build-Umgebung

Das aicas-Team freut sich darüber, Ihnen die neueste Version der JamaicaVM vorstellen zu können. JamaicaVM 2.6 bringt signifikante Verbesserungen in der Benutzung von Java in Echtzeit- und eingebetteten Systemen. Drei Hauptaspekte wurden besonders verbessert: Leistung, Bedienbarkeit, Kompatibilität und OSGi-Unterstützung.

Mit der Größe der Anwendungen, die JamaicaVM nutzen, ist auch der Bedarf für einen schnellen Turnaround im Build-Prozess der Applikation gewachsen. Durch Optimierung an großen Anwendungen mit bis zu 7.000 Klassen und 60.000 Strings war es möglich, zahlreiche Teile von Jamaica zu verbessern. Zum Beispiel werden jetzt Informationen aus Klassendateien direkt im ELF-Format geschrieben. Das Ergebnis ist eine effizientere und robustere Build-Umgebung sowie eine besser ansprechbare Applikation.

Auch die Build-Optionen wurden überarbeitet. Neue Optionen wurden hinzugefügt, um Lazy Linking und erweitertes Profiling zu unterstüt-

zen. Eine verbesserte Übersetzungsanalyse führte dazu, daß einige Optionen jetzt automatisch von Jamaica gesetzt werden. Diese Änderungen spiegeln sich auch im Eclipse-Plugin für die JamaicaVM wieder.

Nicht nur der Builder ist komfortabler geworden, auch wurde die Unterstützung für JDK 1.4 verbessert. Insbesondere Serialisierung und Reflektion wurden anhand der JDK 1.4 Standardimplementierung aktualisiert.

Neben den Standardklassen ist OSGi eine wichtige API für Fahrzeug- und Haustechnik. Es bietet ein flexibles Framework für das dynamische Laden und Ersetzen von Anwendungen während der Laufzeit eines Programmes. JamaicaVM Version 2.6 unterstützt die OSGi-Implementierungen von Prosyst und Frauenhofer IIS.

Eine Evaluierungsversion der JamaicaVM 2.6 ist über die aicas Webseite www.aicas.com erhältlich. Für weitere Informationen über die JamaicaVM kontaktieren Sie bitte info@aicas.com.

↓ Neuigkeiten

Embedded Award 2005

JamaicaVM 2.6 hat den Embedded Award 2005 im Bereich Software gewonnen. Eine unabhängige Jury hat Gewinner in den Kategorien Hardware, Software und Tools gewählt. JamaicaVM gewann, da es die Vorteile der Java-Technologie auch für Entwickler harter Echtzeit- und sicherheitskritischer Systeme nutzbar macht. Die Jury erkannte aicas führende Technologie an.



aicas incorporated

Der amerikanische Markt entwickelt sich zu einem immer wichtigeren Markt für Echtzeit-Java-Technologie. Mit der Gründung von aicas incorporated in New Haven, CT, kann das aicas-Team den US-Markt jetzt besser bedienen. Wir sind erfreut, David P.Reddy in unserem Team begrüßen zu dürfen. Zusätzlich zu dem Support für US-amerikanische und kanadische Kunden ist aicas incorporated auch besonders auf die Anforderungen von Militär und Regierung spezial-

siert. Für Kontaktinformationen siehe www.aicas.com/contacts.html.

JamaicaVM in Siemens SIMOTION

Die Siemens AG hat JamaicaVM in ihre SIMOTION Produkte integriert. Softwareentwickler und Maschinenhersteller können ihre SIMOTION-basierten Systeme nun durch Laden von eigenen Java-Programmen anpassen. JamaicaVM bietet dafür eine sichere Echtzeit-Umgebung, während die Systemleistung und Ansprechbarkeit geschützt wird.

Sicherheitskritisches Java definiert durch HIJA Projekt

Gemeinsames Bestreben von 12 europäischen Organisationen legt Standard fest

Einführung

Der enorme Erfolg von Java beruht auf den vielen Vorteilen gegenüber anderen Programmiersprachen. Beispielsweise erleichtert die wohldefinierte Syntax, ohne einen Präprozessor, die Analyse von in Java geschriebenen Programmen. Die Typisierung und die Indexüberprüfung, zusammen mit der Speicherberei-



nigung, tragen stark zur Robustheit der Programme bei. Darüber hinaus ergibt die Einfachvererbung zusammen mit den Schnittstellen einen guten Kompromiss zwischen Programmrobustheit und -effizienz einerseits und Flexibilität andererseits. Sogar sicherheitskritische Anwendungen in Flugzeugen oder in Fahrzeugsystemen können von diesen Vorteilen profitieren.

Das europäische Projekt HIJA (High Integrity Java(1)), ein Folgeprojekt von HIDOORS(2), AJACS(3), und Espresso(4), hat sich der Herausforderung gestellt, Werkzeuge und Profile zu definieren, um Java in eingebetteten Umgebungen einzusetzen, die ein hohes Maß an Verlässlichkeit erfordern. Beim Start im Juni 2004 taten sich zwölf europäische Organisationen zum HIJA-Konsortium zusammen: aicas, Aonix, Bellstream, Fiat Research Centre, FZI Karlsruhe, Thales-Avionics, Telecom Italia, The Open Group, Trialog, die Universität Karlsruhe, die Polytechnic University of Madrid und die University of York. Eines der Projektziele ist die Definition und Implementierung eines sicherheitskritischen Java-Profils.

Spracherweiterungen wie die Real-Time Specification for Java(6) ermöglichen die Benutzung von Java, wo eine Speicherbereinigung die Ausführung von normalem Java-Code auf unvorhersehbare Weise unterbrechen könnte. Neue Funktionen wie Echtzeit-Threads, die nicht auf den speicherbereinigten Heap zugreifen können, ermöglichen ein vorhersehbares zeitliches Verhalten und können für harte Echtzeit-Aufgaben verwendet werden.

Da bereits Implementierungen der RTSJ verfügbar sind, ist es sinnvoll auf den Safety-Critical-Java (SCJ) Standard der RTSJ aufzusetzen. Dazu begann HIJA im Juni 2004 die Entwicklung des SCJ-Profils für zukünftige, vernetzte und eingebettete Echtzeitsysteme. Dieses Profil bietet eine eingeschränkte Untermenge der RTSJ im Geiste von Ravenscar-Java(6) mit dem Ziel, die Zertifizierung von Systemen mit Java-Code bis zu Level A des DO178B-Standards zu erreichen. Ergebnisse des Profils beinhalten Thread-Modelle, den Synchronisationsmechanismus, Speichermodelle und Annotationen für die rechnerunterstützte Korrektheitsüberprüfung. Ein Entwurf dieses Profils wurde bei der Open Group als Basis für den bevorstehenden Safety-Critical-Java Standard eingereicht.

Das Projekt definiert auch Werkzeuge, um das SCJ-Profil zu vervollständigen. Verifikationswerkzeuge werden entwickelt, um die funktionale und nicht-funktionale (Ausführungszeit, Speicherbenutzung, etc.) Korrektheit von Anwendungen zu prüfen. Diese Werkzeuge werden dazu beitragen, die Robustheit von SCJ-Programmen zu sichern.

Richtlinien

Die SCJ-Definition von HIJA hat die Erstellung eines Profils zum Ziel, welches die Vorteile von Java wie Portabilität, Interoperabilität, objektorientiertes Design und die vielen Werkzeuge und Bibliotheken, die für Java erhältlich sind, beibehält und gleichzeitig eine Untermenge der Sprache bereitstellt, die einfach genug ist, um von statischen Werkzeugen analysiert und für sicherheitskritische Systeme zertifiziert werden zu können. Drei tragende Prinzipien wurden zu Richtlinien für SCJ erklärt.

- SCJ soll konservativ sein: keine der sicherheitskritischen Gemeinschaft fremden Konzepte sollen eingebunden werden.
- SCJ soll RTSJ-konform sein: SCJ soll eine Untermenge der RTSJ sein und nur wenige neue Funktionen bieten. Selbst diese wenigen neuen Funktionen müssen kompatibel zum Standard-Java und der RTSJ sein.

- SCJ soll Java-Metadaten benutzen: Metadaten-Annotationen, obwohl nicht Teil der RTSJ an sich, sind Teil von Java und können daher benutzt werden, um die Offline-Analyse und die Code-Dokumentation in SCJ zu erleichtern.

HIJA Safety-Critical-Java Profil

Das HIJA SCJ-Profil verlangt gravierende Einschränkungen in Java-Programmen. Es gibt keine Bestimmung über Speicherbereinigung im Profil. Die Ausführung wird in zwei getrennte Phasen aufgeteilt: Initialisierungsphase und Missionsphase. Während der Missionsphase sind dynamisches Laden, Thread-Erzeugung und das Anlegen von Objekten in globalen Speicherbereichen nicht erlaubt. Stattdessen werden temporäre Objekte in lokalen Speicherbereichen für jede periodische und asynchrone Aufgabe benutzt. Annotationen werden verwendet, um zu dokumentieren welche Methoden oder Klassen in der Missionsphase sicher benutzt werden können.

Nebenläufigkeitsmodell

Das SCJ-Profil verwendet das Nebenläufigkeitsmodell der RTSJ, d. h. das sogenannte "fixed priority preemptive scheduling", dass 28 verschiedene Prioritätsstufen und FIFO-Ausführung für Aufgaben gleicher Priorität benutzt.

Synchronisation

Java bietet mit den Monitoren als Teil aller Java-Objekte einen leistungsfähigen Synchronisationsmechanismus. Die RTSJ hat diese Monitore für den Einsatz in Echtzeitsystemen um Prioritätsvererbungs- und Prioritätsanpassungs-Protokolle (*priority inheritance* und *priority ceiling emulation*) erweitert. Diese beiden Synchronisationsprotokolle vermeiden die Prioritätsinversion. Für das SCJ-Profil muss es möglich sein zu überprüfen, dass keine synchronisationsbedingten Fehler in einer Anwendung vorhanden sind. Denkbare Fehler sind potentielle dead-locks, Prioritätsinversion und Wettlaufsituationen. Sowohl Prioritätsvererbung als auch Prioritätsanpassung (*priority ceiling emulation*) vermeiden die Prioritätsinversion. Jedoch kann

Prioritätsvererbung keine dead-locks verhindern, wenn verschachtelte Monitore benutzt werden, und der Thread blockiert während er eine Semaphore hält. Aus diesem Grund benutzt das SCJ-Profil nur das *priority ceiling emulation*-Protokoll.

Speicherverwaltung

Im SCJ-Profil wird keine Speicherbereinigung benutzt. In der RTSJ-Formulierungen bedeutet das, dass Objekte nicht im Heap-Speicher angelegt werden, da der Heap unter der Kontrolle der Speicherbereinigung steht. Stattdessen werden alle Objekte in SCJ entweder dauerhaft oder in sicheren, bereichsbasierten Speicherbereichen angelegt. Das wiederholte Anlegen von dauerhaften Objekten führt irgendwann dazu, dass das System keinen verfügbaren Speicher mehr hat. Daher wird das Anlegen solcher Objekte auf die Initialisierungsphase beschränkt. Alle Objekte, die von verschiedenen *schedulable objects* geteilt werden, müssen in der Initialisierungsphase statisch dauerhaft angelegt werden. Sobald die Missionsphase begonnen hat, bleibt die Menge an belegtem Speicher konstant. Das vermeidet Fehler aufgrund unzureichenden Speichers während der Missionsphase.

Das Anlegen von temporären Objekten kann jedoch während der Missionsphase nicht verhindert werden, ohne den objektorientierten Programmierstil von Java zu opfern. Für das Anlegen von temporären Objekten während der Missionsphase erhält jedes dieser *schedulable objects* seinen eigenen Speicherbereich (*scoped memory region*). Eine wiederholte Verwendung (Schachtelung) solcher Speicherbereiche ist nicht erlaubt. Die Zuweisungsregeln der RTSJ für *scoped memory regions* garantieren dann, dass diese temporären Objekte nur von einem *schedulable object* benutzt werden dürfen.

Der Gültigkeitsbereich eines *schedulable objects* wird automatisch bei jedem Release betreten und nach Ausführung des Releases verlassen. Alle Objekte, die während einem Release erzeugt wurden, werden vor dem nächsten Release wiederverwendet. Da jeder Rahmen lokal zu einem *schedulable object* ist, wird das Anlegen temporärer Objekte in einem Task nicht durch andere Tasks beeinflusst.

Mit diesem Speichermodell muss durch eine statische Analyse überprüft werden, dass keine Zuweisungsfehler zur Laufzeit auftreten können. Ein Zuweisungsfehler tritt auf, wenn eine Referenz zu einem Objekt, das im *scoped memory* erzeugt wurde, einer statischen Variablen oder einem Objekt zugewiesen wird, welches in einem Speicherbereich erzeugt wurde, der eine längere Lebensdauer besitzt, wie Heap-, dauerhafter Speicher oder ein umgebender *scoped memory*. Da Heap-Speicher im Profil nicht benutzt wird und Verschachtelung von *scoped memory* nicht erlaubt ist, muss die statische Analyse nur nachweisen, dass es keine Zuweisungen temporärer Objekte zu statischen Variablen oder Objekten im dauerhaften Speicher gibt.

Korrektheitsanalysewerkzeuge

Eine ganze Reihe von Werkzeugen für verschiedene Ebenen der Verifikation sind innerhalb des HIJA-Projektes in Entwicklung. Das Werkzeug Key benutzt Annotationen und wird dem Entwickler Verifikationsergebnisse für funktionale Korrektheit bieten.

Eine programmweite Datenflußanalyse (DFA), wird Kontroll- und Datenflußinformationen sammeln, die von Werkzeugen für die Analyse nicht-funktionaler Korrektheit benötigt werden.

Die maximale Ablaufzeitanalyse (MAZA) Verwendet die Ausgabe der DFA, den ausführbaren Code und zusätzliche Indexinformationen, um die maximale Ausführungszeit jeder Aufgabe zu bestimmen.

Auf ähnliche Weise bestimmt das Werkzeug zur Speicherbedarfsanalyse die maximale Heap- und Stackgröße.

Code Consistency analysis überprüft die Anwendung auf bestimmte Fehlerbedingungen wie Laufzeitfehler (Zuweisungen, Typumwandlungen, etc.) und dead-locks.

Annotationen im Java-Code können durch das Werkzeug *Annotation Check* geprüft werden, welches auf der DFA basiert.

Der *Model Generator* schließlich verwendet Klassendateien, Indexinformationen von Annotationen und die Ergebnisse der MAZA und *Memory Usage Analysis*, um ein Modell zu generieren, das vom

↓ Veranstaltungen

Besuchen sie aicas auf den folgenden Messen:

Java & (RT) Embedded Systems, Leuven

19. Mai 2005

Wind River 2005 Worldwide User Conference, Orlando

22.-25. Mai 2005

Java Forum 2005, Stuttgart

7. Juli 2005

Embedded Technology, Yokohama

14.-18. November 2005

UPPAAL Modellprüfer verifiziert werden kann.

Alle Analysewerkzeuge liefern dem Entwickler Informationen, um Quelltext- oder Annotationsveränderungen zu erleichtern.

Schlussfolgerung

Die von der RTSJ definierten Funktionen ermöglichen die Nutzung von Java in neuen Einsatzgebieten mit strengen Anforderungen an das zeitliche Verhalten der Anwendungen. Jedoch erfordert der Einsatz in sicherheitskritischen Anwendungen gravierende Einschränkungen der Funktionalität, um die statische Analyse und Zertifizierung zu ermöglichen. Das HIJA-Projekt bietet ein sicherheitskritisches Java-Profil und entwickelt Werkzeuge zur Korrektheitsüberprüfung von Anwendungen. Dies ist ein großer Schritt in Richtung der Zertifizierung von Java für sicherheitskritische Anwendungen. Dadurch werden die Produktivitäts- und Sicherheitsvorteile durch die Nutzung von Java-Technologie in unterschiedlichen Anwendungen für die Entwicklung sicherheitskritischer Anwendungen verfügbar sein.

Ausblick

Die nächste Ausgabe wird Echtzeit-CORBA in Java vorstellen.

- (1) <http://www.hija.info>
- (2) <http://www.hidoors.org>
- (3) <http://www.ajacs.org>
- (4) <http://www.irisa.fr/rntl-expresso/docs/hip-api.pdf>
- (5) <https://www.rtsj.dev.java.net>

Das IST-Project HIDOORS wurde erfolgreich abgeschlossen

Java-Technologie in Echtzeitsystemen durch bessere Implementierungen und Werkzeuge

Java ist für Desktop- und Internetanwendungen immer wichtiger geworden. Verbesserte Portabilität, Typsicherheit und Syntax zählen zu den wichtigsten Vorteilen. Mit geeigneten Werkzeugen und Technologien können diese Vorteile auch in Echtzeit- und eingebetteten Systemen genutzt werden. Das High Integrity Distributed Object-Oriented Realtime Systems (HIDOORS) Projekt hat einen bedeutenden Beitrag dazu geleistet.

Unter Federführung des IST-Programms hat das HIDOORS-Projekt sechs europäische Partner zusammengebracht, um den Stand der Echtzeitprogrammierung mit Java in Form von JamaicaVM zu verbessern. Dieses Projekt beinhaltete nicht nur Anstrengungen die grund-

legende Java-Technologie zu verbessern, sondern auch Werkzeuge zu schaffen, die nötig sind, um Java im weiten Feld eingebetteter Echtzeitanwendungen einzusetzen. Vier der Partner waren Technologie-lieferanten: die Universität Linköping für Modellüberprüfung, Aonix für UML-Designwerkzeuge, FZI für Netzwerke und Analyse und aicas für Java-Technologie. Die anderen beiden Partner, R.O.S.E. Informatik und Skysoft, lieferten Validierungsprojekte zusammen mit Aonix.

Bedeutende Aspekte des Projekts waren Echtzeit-UML-Modellierung, MDA-Code-Generierung, *Model Driven Verification* (MDV) mit Modellüberprüfung auf UML-Ebene, eine Java-Implementierung auf dem neuesten Stand der Technik mit deterministischer Speicherbereinigung und einer kompletten Implementierung der RTSJ, ein leichtgewichtiger Ereignisdienst und maximale Ablaufzeitanalyse (MAZA) für statisch übersetzten Java-Code.

Bemerkenswerte Ergebnisse wurden bei der Verfeinerung der Technik und der Verbesserung der kommerziellen Produkte erzielt. Das aicas-Team mit ihrer deterministischen JamaicaVM und Aonix mit ihrem UML-Entwicklungswerkzeug Ameos konnten ihre Produkte durch HIDOORS-Forschungs- und Entwicklungsarbeit substantiell verbessern. Professor Uwe Aßmann leitete die Arbeit in Linköping, um die Konzepte von MDA auf Verifikation nach dem Gedanken der MDV auszuweiten. Die zentrale Idee ist, MDA-Konzepte nicht nur zur Generierung von Code, sondern auch von Modellabstraktionen zur Verwendung in Analysewerkzeugen wie dem UP-PAAL Modellprüfer zu benutzen. Das FZI konnte die Erkenntnisse mit MAZA auf Java anwenden und Netzwerkerfahrungen auf verteilten Echtzeitsystemen gewinnen. Natürlich wäre das Projekt ohne Erfahrungen mit realen Problemen in den Bereichen der Fahrzeug- und Seefahrttechnik sowie in der zivilen Flugtechnik kein Erfolg geworden.

Die Entwicklung einer kommerziellen Java-Laufzeitumgebung und eines verbesserten statischen Übersetzers waren die wichtigsten Projektaufgaben für das aicas-Team. JamaicaVM ist in Folge dessen schneller, vollständiger und robuster geworden. Das HIDOORS-Projekt war ein wichtiger Eckpfeiler, um aicas im Markt zu etablieren. Nichtsdestoweniger profitiert Java-Technologie von der Entwicklung von Werkzeugen zum Design und zur Analyse von Echtzeitsystemen.

HIDOORS hat gezeigt dass eine Gruppe von kleinen aber motivierten Unternehmen den neuesten Stand der Technik signifikant beeinflussen kann. Java-Technologie ist jetzt besser für die Bedürfnisse des Marktes für eingebettete Systeme angepasst.



HIDOORS

Gute Ansprechbarkeit mit JamaicaVM

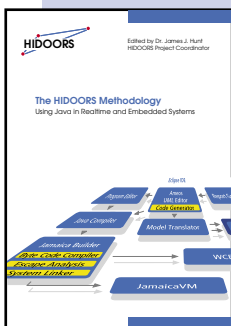
Tests zeigen stabile periodische Ansprechbarkeit bis 150 µsec

Deterministische Ansprechbarkeit, sogar für Threads, die dynamisch Speicher benötigen, ist ein großer Vorteil der Speicherverwaltung von JamaicaVM. In der letzten aicas news wurde ein Test mit sehr kleiner Schwankung mit einer Periode von 1 ms auf einem 300-MHz-PPC mit einer Echtzeituhr der Genauigkeit 500 µsec präsentiert. Neue Tests mit Version 2.6 zeigen, dass noch kürzere Perioden erreicht werden können. Tests auf einem 2.8-GHz-Xeon betrieben mit Redhawk Linux ergaben eine Standardabweichung der Schwankung von 2 µsec in einer Periode von 150 µsec. Die maximale Schwankung betrug 70 µsec. Die ordnungsgemäße Abstimmung mit einem Echtzeitbetriebssystem könnte gegebenenfalls dasselbe periodische Zeitverhalten mit noch besserem Ergebnissen liefern.

↓ Neues Buch

The HIDOORS Methodology Using Java in Realtime and Embedded Systems

Herausgeber: Dr. James J. Hunt



Die HIDOORS Entwicklungsgruppe baut auf verbreiteten Paradigmen moderner Softwaresysteme auf: Modellierung mit UML und plattformunabhängiges, objektorientiertes Programmieren mit Java. Dieses Buch behandelt die wichtigsten Aspekte der Nutzung von Java in Echtzeit- und eingebetteten Systemen. Das HIDOORS-Team vermittelt Hintergrundwissen von Programmierung über UML-Modellierung bis hin zu Analyse und Tests. Die gesamte Werkzeugkette wird im Zusammenspiel mit dem System besprochen. Die Theorie, Regeln und Randbedingungen von HIDOORS bilden eine Methodik für Echtzeit- und sicherheitskritisches Programmieren. Neue Benutzer von JamaicaVM werden unschätzbare Informationen finden, wie sie Java für ihre Echtzeit- und sicherheitskritischen Projekte einsetzen können.

↓ Kontakt

aicas GmbH

Haid-und-Neu-Straße 18
D-76131 Karlsruhe
Germany

tel +49 721 663 968-0
fax +49 721 663 968-99
email info@aicas.com
web www.aicas.com