Gesellschaft für Informatik
Fachgruppe 3.1.4
Betriebssysteme
http://www13.in.tum.de/gifgbs

Der Sprecher
Prof. Dr. W. Schröder-Preikschat

Otto-von-Guericke-Universität
Magdeburg
Fakultät für Informatik
Universitätsplatz 2
39106 Magdeburg

Tel.: (0391) 67 18829 / 18664
Fax: (0391) 67 12810

E-Mail: wosch@cs.uni-magdeburg.de

Einladung

zum Frühjahrstreffen
am 02./03. März 2000 in Erlangen

Wie zertifiziert der TÜV ein Betriebssystem? Welche Probleme waren bei der Entwicklung des OSEK-basierten Betriebssystems für eine neue KFZ-Modellreihe zu bewältigen? Kopfhörer, Armbanduhr und Mobiltelefon in einem Netzwerk - welche Herausforderungen stellen solche Szenarien an Netzwerk und Betriebssystem?

Über diese Fragen und natürlich über eine ganze Reihe weiterer aktueller Forschungsarbeiten auf dem Gebiet Betriebssysteme in Unternehmen und an den Universitäten in Deutschland wird auf dem diesjährigen Frühjahrstreffen der Fachgruppe Betriebssysteme unter dem Thema

Betriebssysteme jenseits von PC und Workstation:
Automation, Embedded Systems, Mobile Computing

berichtet. Außerdem wird im Vorfeld des Treffens am 01. März ein Tutorial über Architektur und Umsetzung von Betriebssystemkonzepten in Windows NT angeboten.
Unser Aufruf zu Vortragseinreichungen hat diesmal wieder ein erfreulich großes Echo gefunden, so dass wir ein interessantes Vortragsangebot zusammenstellen konnten, zu dem wir Sie herzlich einladen!

Zur Teilnahme am Treffen der Fachgruppe füllen Sie bitte das beiliegende Anmeldeformular bzw. das Formular auf der Informationsseite im WWW aus und senden es bis spätestens 20. Februar 2000 an den lokalen Organisator

Dr. Jürgen Kleinöder
Universität Erlangen, Informatik 4
Martensstr. 1
91058 Erlangen

Tel.: (09131) 85 - 2 80 28 / 2 72 77
Fax: (09131) 85 - 2 87 32
E-Mail: jk@informatik.uni-erlangen.de

Informationsseite: `http://www.informatik.uni-erlangen.de/fgbs/`

Tagungsort: Universität Erlangen
Technische Fakultät (Universität Südgelände)
(Martensstr. 1 / Erwin-Rommel-Str. 60)
Mensa-/Hörsaalgebäude, Konferenzsaal K1

Beginn: 02.03.2000, 11:15 Uhr
Ende: 03.03.2000, 13:30 Uhr

Hotelreservierungen müssen selbst vorgenommen werden. Weitergehende Hotelinformationen und An- und Abreisemöglichkeiten finden Sie über die Informationsseite im WWW. Herr Dr. Kleinöder oder Herr Dr. Bellosa (Tel. (09131) 85 - 28820) helfen Ihnen bei Fragen gerne weiter.

Programm

Donnerstag, 02.03.2000

11:15 - 11:30 Begrüßung

Sitzung 1: OSEK

11:30 - 12:05 **ProOSEK: Vom OSEK/VDX-Standard zum Echtzeit-Betriebssystem**
Jochen Schoof, 3SOFT GmbH, Erlangen

12:05 - 12:40 **LEO/P4 - A -Kernel Based OSEK Implementation**
Robert Kaiser, Sysgo Real-Time Solutions GmbH, Klein-Winternheim

12:40 - 13:45 ***Mittagessen***

Sitzung 2: Betriebssysteme für mobile Systeme

13:45 - 14:20 **Ein Vergleich von unterschiedlichen Betriebssystemen für sichere mobile Endgeräte**
Harald Görl, Claudia Eckert, Uwe Baumgarten, Technische Universität München

14:20 - 14:55 **CoolIX und EndurIX- Prozeßspezifische Drosselung der Systemaktivität zur Begrenzung der Leistungsaufname und des Energieverbrauchs**
Frank Bellosa, Universität Erlangen, Lehrstuhl für Betriebssysteme

14:55 - 15:30 **DEAPspace: Transient Ad-Hoc Networking of Pervasive Devices**
Dirk Husemann, IBM Research Division, IBM Zurich Research Laboratory

15:30 - 16:00 ***Kaffeepause***

Sitzung 3: Betriebssystemarchitekturen

16:00 - 16:35 **Generative Maßschneiderung von Betriebssystemen**
Martin Becker, Lothar Baum, Lars Geyer, Georg Molter, AG Systemsoftware, Universität Kaiserslautern

16:35 - 17:10 **The SawMill Project**
Jochen Liedtke, Universität Karlsruhe

17:10 - 17:45 **Einfluss von Cache-Coloring auf die Vorhersagbarkeit und Performance von echtzeitfähigen Netzwerkkomponenten**
Jork Löser, Hermann Härtig, TU Dresden

17:45 - 18:30 **Diskussionsrunde: Zukunft der Betriebssystemforschung in Deutschland und Stellung der GI-Fachgruppe 3.1.4 Betriebssysteme**

19:15 ***Abendveranstaltung***

Freitag, 03.03.2000

Sitzung 4: Komponententechnologie und Betriebssysteme

08:30 - 09:05 **Komponierbarkeit eingebetteter Echtzeitsysteme**
Jan Richling, Humboldt-Universität zu Berlin, Institut für Informatik

09:05 - 09:40 **Komponentenbasierte Softwareentwicklung für Eingebettete Echtzeitsysteme**
Uwe Rastofer, Universität Erlangen, Lehrstuhl für Betriebssysteme

09:40 - 10:15 **Systemerstellung unter dem Echtzeitbetriebssystem Ambrosia/MP**
Ingo Stierand, Universität Oldenburg, Abteilung Betriebssystem und Verteilte Systeme

10:15 - 10:45 ***Kaffeepause***

Sitzung 5: Sicherheitskritische Systeme

10:45 - 11:20 **Protokolle zur internen Uhrensynchronisation in sicherheitskritischen Echtzeitsystemen**
Horst F. Wedde, Wolfgang Freund, Informatik III, Universität Dortmund

11:20 - 12:20 **Evaluierung von sicherheitsgerichteten Betriebssystemen**
Rainer Faller, TÜV Product Service GmbH, München

12:20 - 13:20 **Sitzung der Fachgruppe**

13:20 ***Mittagessen***

Tutorial

Traditionsgemäß plant die Fachgruppe “Betriebssysteme” der GI auch zum Frühjahrstreffen in Erlangen wieder ein Tutorial:

Tutorial "Windows NT (WINT)"

Wann? Mittwoch, 01. März 2000, 13:00 - 17:00

Wo? Universität Erlangen
Technische Fakultät
Martensstr. 1 (Rechenzentrumsgebäude), Seminarraum 0.031

Wer? Winfried Kalfa
Prof. Dr.
TU Chemnitz
Fakultät für Informatik, Betriebssysteme
Post: 09107 Chemnitz
Sitz: Straße der Nationen 62, Zimmer 351c
Tel.: (0371) 531-1715/...-1430, Fax.: ...-1530
E-Mail: kalfa@informatik.tu-chemnitz.de
WWW: http://www.tu-chemnitz.de/informatik/osg

Was? Das Betriebssystem (BS) Windows NT wird aus Sicht eines "Betriebssystemlers" betrachtet, welche Wirkprinzipien von BS in WINT wie realisiert werden. Im einzelnen sollen diskutiert werden:

- Architektur
- Prozeßmodell
- Programmierplattformen
- Verwaltung von Aktivitäten und Betriebsmitteln
- Synchronisation und interne Kommunikation
- WINT im Netz
- Domänen
- Client - Server
- WINT und UNIX/Linux
- Dateisystem
- Sicherheit und Schutz
- Windows 2000.

Material: Die Teilnehmer erhalten die Folien in Papierform.

Teilnehmergebühr: Wahrscheinlich 40,00 DM, Kassierung in bar vor Ort.

Anmeldung: Benutzen Sie bitte das angehängte Formular und senden Sie es bis 15.02.2000 an eine der drei oben angegebenen Zieladressen (E-Mail bevorzugt).

Anmeldung zum Tutorial "Windows NT" am 1.3.00 in Erlangen

Name: .....

Vorname: .....

Institution: .....

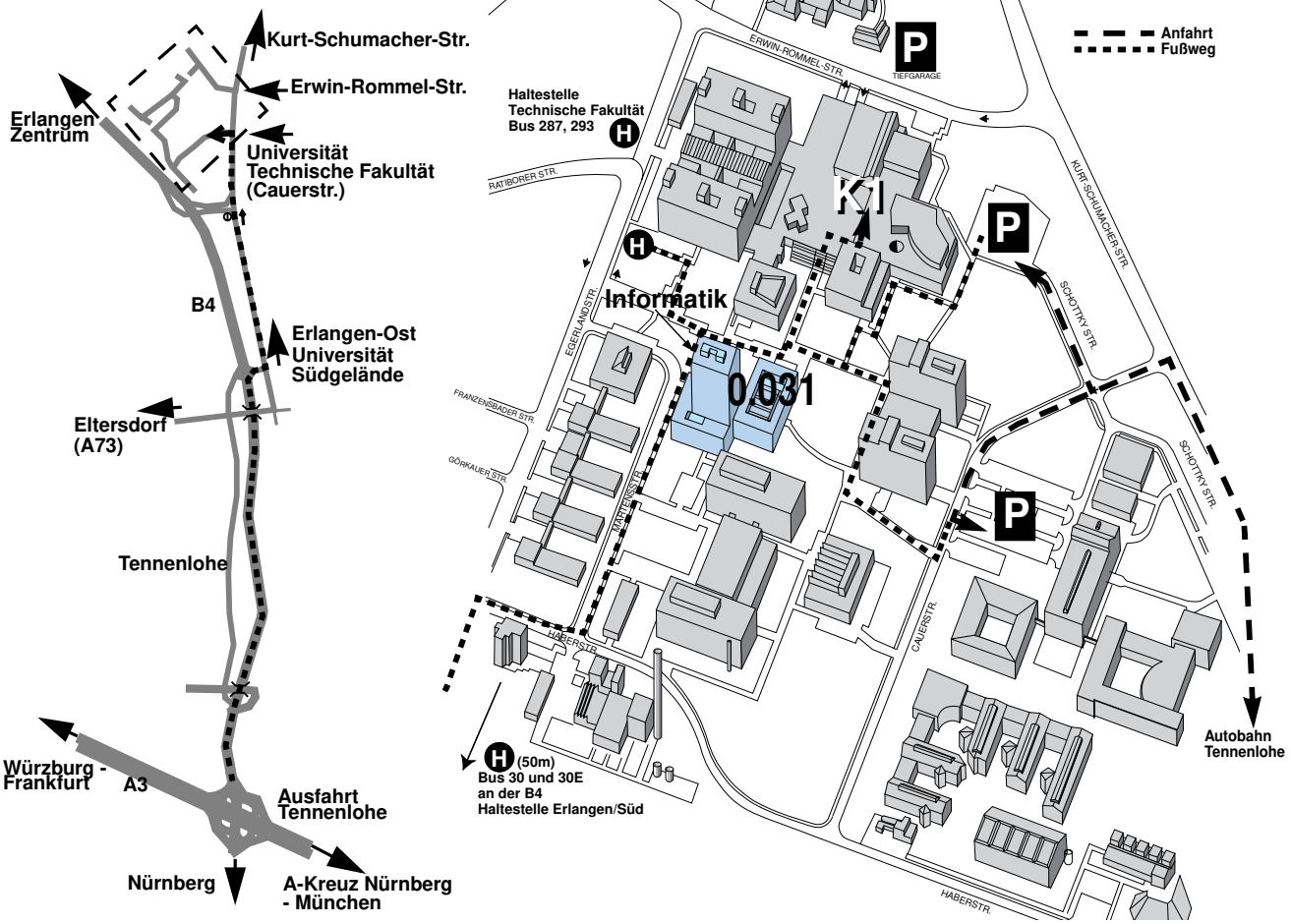
Adresse: .....

Telefon: .....

Telefax: .....

E-mail: .....

Anreiseinformationen



Mit der Bahn:
Ankunft am Bahnhof Erlangen Hbf. Von dort haben Sie die Wahl zwischen Bus und Taxi:
Bus (ca. 20min): Steigen Sie in den Bus Nr. 287 in Richtung Sebaldussiedlung. An der Haltestelle Technische Fakultät steigen Sie aus. Beachten Sie im Fahrplan, daß viele Busse nicht während der vorlesungsfreien Zeiten verkehren. Diese sind vom 1. März bis 30. April und vom 1. August bis zum 31. Oktober.
Taxi (ca. 15min): Lassen Sie sich in die Martensstr. 1 fahren.

Da wir davon ausgehen, dass ein grosser Teil der Teilnehmer mit dem IC 818 (ab Nürnberg 10:34 - an Erlangen 10:47) ankommen wird, planen wir für den Transport vom Erlanger Bahnhof zum Uni-Gelände ca. 10 Minuten nach Eintreffen des IC 818 einen Shuttle-Dienst. Bitte geben Sie bei der Anmeldung an, ob Sie diesen nutzen möchten.

Mit dem Auto:
Über die Autobahn A3 aus Richtung München oder Würzburg: Nehmen Sie die Ausfahrt Tennenlohe und fahren Sie die B4 Richtung Erlangen. Nehmen Sie dann die Ausfahrt Universität Südgelände. Nach der Ausfahrt links ab, Richtung Erlangen bzw. Südgelände (Kurt-Schumacher-Str.). Fahren Sie geradeaus bis Sie links in die Cauer- oder Erwin-Rommel-Straße einbiegen können (siehe Detailkarten).

Bus aus dem Stadtgebiet Nürnberg:
Mit Straßenbahnlinie 9 (aus Ri. Hauptbahnhof, Rathenauplatz) oder Straßenbahnlinie 4 (aus Ri. Plärrer) bis zur Endhaltestelle Thon. Dort umsteigen in Omnibuslinie 30 oder 30E bis Haltestelle Erlangen/Süd. Den Fußweg nach rechts gehen. Nach ca. 50m trifft man auf die Einmündung Egerlandstr./Haberstr. Über Haberstr.-Martensstr. oder Egerlandstr.-Martensstr. laufen (siehe Detailkarte rechts).

Kurzfassungen der Vorträge

ProOSEK: Vom OSEK/VDX-Standard zum Echtzeit-Betriebssystem

Jochen Schoof

3SOFT GmbH, Wetterkreuz 19a, 91058 Erlangen
Jochen.Schoof@3SOFT.de

Der OSEK/VDX-Standard[1] definiert ein Betriebssystem für den Einsatz in embedded real-time Geräten. Im wesentlichen wird dabei aber nur eine Programmierschnittstelle (API) beschrieben. Zu den für den Anwender wichtigen Fragen des Zeit- und Ressourcenverbrauchs äußert sich der Standard nicht. Folglich spielt für die Konformität einer Implementierung weder ihr zeitliches Verhalten noch ihr Umgang mit den Ressourcen eine Rolle.

Im Automobilsektor als dem Haupteinsatzgebiet von OSEK/VDX-konformen Systemen werden überwiegend 16-Bit Microcontroller in Single-Chip Konfiguration benutzt. Typische Speichergrößen bewegen sich zwischen 32 und 128 KB ROM bzw. zwischen ein und zehn KB RAM. Die Prozessoren werden mit Frequenzen zwischen vier und acht Megahertz getaktet, wobei man oft gezwungen ist, niedrigere Werte einzuhalten, um die Abstrahlung möglichst gering zu halten. Hinzu kommt, dass neben dem Betriebssystem noch weitere standardisierte Module zu integrieren sind (z.B. Kommunikation, Diagnose). Die Optimierung des Betriebssystems im Hinblick auf Zeit- und Ressourcenverbrauch ist also von zentraler Bedeutung.

Da OSEK/VDX eine individuelle Konfiguration des Betriebssystems für jede Applikation vorsieht, kann bei dieser Optimierung auch Wissen über deren genaue Anforderungen genutzt werden. Entscheidend hierfür ist ein Konfigurationswerkzeug, das in der Lage ist, die jeweiligen Optimierungspotenziale zu erkennen und auszunutzen. Nur so können ohne die direkte Beteiligung des Anwenders Verbesserungen erreicht werden.

Ein gutes Beispiel für das vorhandene Optimierungspotenzial ist der Scheduler: Der Bogen spannt sich von Applikationen mit einer einzigen single-shot Task bis zu solchen mit einer Vielzahl eventgesteuerter, präemptiver und mehrfach aktivierbarer Tasks. Zwischen diesen beiden Extremen mit sehr unterschiedlichen Anforderungen gibt es eine ganze Reihe sinnvoller Abstufungen. So definiert OSEK/VDX bereits vier so genannte Conformance Classes und drei grundlegende Scheduling-Strategien, die eine Skalierung in zwölf Schritten erlauben. In der Praxis zeigt sich aber, dass sogar eine noch feinere Anpassung möglich ist. So ist zum Beispiel das vom Standard hergestellte Junktim zwischen mehrfach aktivierbaren Tasks und mehreren Task gleicher Priorität in vielen Fällen nicht besonders sinnvoll.

Am Beispiel von ProOSEK, einer in der Automobilindustrie bereits im Einsatz befindlichen OSEK-Implementierung, werden mögliche Strategien zur Optimierung des OSEK-Kernels vorgestellt. Als Ergebnis des Einsatzes dieser Techniken sind minimale Kernelgrößen von weniger als einem KB erreichbar.

1. Kostenlos zu beziehen unter der URL `http://www.osek-vdx.org/osekvdx_os.html`.

LEO/P4 - A µ-Kernel Based OSEK Implementation

Robert Kaiser
Sysgo Real-Time Solutions GmbH, Klein-Winternheim. Germany
rkaiser@sysgo.de
January 18, 2000

1. Abstract

This article introduces LEO/P4, an implementation of the standardized OSEK OS API as a server on top of the new µ-kernel P4. Together with the OSEK emulation environments LEO/Lynx or LEO/Linux, it provides an OSEK run-time environment that can seamlessly support OSEK applications from the early stages of development unto the final production version in a vehicle.

The P4 µ-kernel is a new, advanced re-implementation of the L4 µ-kernel as specified by Liedtke /4/. It features a processor independent API and is internally structured for better portability. Platform specific support routines are provided to P4 by an external module (PSP), so porting P4 to another platform within the same processor family can be achieved without modifications to the µ-kernel's binary code.

In contrast to traditional monolithic implementations of OSEK, the µ-kernel-based approach allows for multiple operating system APIs and instanciations to coexist. Specifically, it is possible to have multiple instances of OSEK OS running independently in a single machine, each one in it's own protected address space.

Moreover, a Linux server can also be added to run in parallel with the OSEK OS instance(s), thus turning the system into a full-featured UNIX workstation. This does not affect OSEK's real-time characteristics, so the resulting system is a very comfortable and efficient environment for developing, debugging and testing OSEK-based code under real-world conditions.

All interaction between the servers is based on the µ-kernel's inter process communication (IPC) mechanism, which features the capability of transparent re-routing of messages across a network. Therefore, servers can be migrated from the development system to the embedded controller without any modification to their code. This not only provides an easy way to implement distributed systems, it also makes the process of moving OSEK applications from the development system to their designated target ECU far less painful than usual.

We will start with brief introductions of the OSEK OS standard and of LEO, SYSGO's OSEK OS implementation and will then focus on the benefits of the µ-kernel-based implementation, as well as on the P4 µ-kernel itself.

Ein Vergleich von unterschiedlichen Betriebssystemen für sichere mobile Endgeräte

Harald Görl, Claudia Eckert, Uwe Baumgarten
Technische Universität München
{goerl, eckertc, baumgaru}@in.tum.de

Überblick

Im Rahmen des DFG-Schwerpunktprogramms „Sicherheit in der Informations- und Kommunikationstechnik“ wird in dem Projekt „LUCA“ sichere Betriebssoftware für mobile Endgeräte entwickelt. In diesem Vortrag wird zunächst die Vorgehensweise dabei kurz erläutert. Im zweiten Teil wird ein Vergleich zwischen drei existierenden Betriebssystemen präsentiert, wobei vor allem Sicherheitsaspekte und Tauglichkeit für den Einsatz als mobile Betriebssoftware berücksichtigt werden.

Vorgehen

Für die Entwicklung eines Konzeptes werden zunächst existierende Systeme untersucht. Die Betriebssoftware soll unter anderem Mechanismen zur sicheren Authentifizierung der Benutzer und zur sicheren Übertragung von Daten besitzen. Dazu wird auf bestehende Verfahren zurückgegriffen. Weitere Forderungen sind die dynamische Erweiterbarkeit des Kernels und die Personalisierbarkeit für unterschiedliche Benutzer.

Ein Vergleich: EPOC, PalmOS, Linux

Die drei Betriebssysteme EPOC, PalmOS und Linux können als Grundlage für die Realisierung einer sicheren mobilen Systemsoftware dienen und werden deshalb untersucht und anschließend miteinander verglichen.
EPOC und PalmOS sind Systeme, die speziell für den mobilen Bereich entwickelt wurden. Dabei gehören ein einfaches Speichermanagement, eine grafische Bedienoberfläche, Schnittstellen für stiftbasierte Eingabe und verschiedene Kommunikationsschnittstellen zur Standardausstattung. Diese Systeme sind so konzipiert, daß sie möglichst wenig Ressourcen verbrauchen und die Entwicklung für grafische Anwendungen vereinfachen. Dabei werden meist keine wirkungsvollen Speicherschutzmaßnahmen oder differenzierte Zugriffskontrollen angewendet.
Trotz vieler Unterschiede haben die bekannten Systeme für mobile Endgeräte gemeinsam, daß sie nur auf einen Benutzer zugeschnitten sind und praktisch keine wirkungsvollen Sicherheitsmechanismen bieten. Das Anmelden verschiedener Benutzer ist auf den meisten Systemen überhaupt nicht möglich. Genau diese Eigenschaften besitzen aber auf Linux basierende mobile Betriebssysteme. Außerdem sind Quelltexte und ausreichend Dokumentation zur Beschreibung des Linux-Kerns vorhanden, so daß dieser als Basis für eine sichere mobile Systemsoftware in Frage kommt.

CoolIX und EndurIX

Prozeßspezifische Drosselung der Systemaktivität zur Begrenzung der Leistungsaufname und des Energieverbrauchs

Frank Bellosa

Universität Erlangen-Nürnberg
Lehrstuhl für Betriebssysteme

Kompakte und portable Rechner stellen hohe Anforderungen hinsichtlich einer geringe Leistungsaufnahme und einer lange Betriebsdauer.

Computer mit passiver Kühlung und bergrenzten Möglichkeiten der Wärmeableitung können durchaus für eine kurze Zeit eine hohe Leistung aufnehmen, wenn die Energie danach langsam wieder abgegeben werden kann. Im Rahmen des CoolIX-Projekts wird die Leistungsaufnahme einzelner Prozesse gemessen und analysiert, um bei Überschreitung der Wärmeleitungskapazität die Systemaktivität zu drosseln. Damit können kompakte Geräte im interaktiven Betrieb kurzfristig eine hohe Rechenleistung anbieten.
(siehe auch http://www4.cs.fau.de/CoolIX)

Ein vielversprechender Ansatz, um die Zahl der Instruktionen zu maximieren, die ein Prozessor mit einer gewissen Energiemenge ausführen kann, ist die Reduktion der Taktfrequenz bei gleichzeitiger Absenkung der Versorgungsspannung. Ziel des EndurIX-Projektes ist es, durch eine prozeßspezifische Abarbeitungsgeschwindigkeit den Energieverbrauch zu reduziert, ohne das Echtzeitverhalten des Gesamtsystems zu verschlechtern. Die Betriebsdauer vieler batteriebetriebener Geräte kann so entscheident verlängert werden können.
(siehe auch http://www4.cs.fau.de/EndurIX)

DEAPspace: Transient Ad-Hoc Networking of Pervasive Devices

Reto Hermann Dirk Husemann Michael Moser Mike Nidd
Christian Rohner Andreas Schade

26th January 2000

1 Project Goals

The aim of IBM Research's DEAPspace project is to connect mobile and pervasive devices in transient ad hoc networks to allow available resources to be utilized and coordinated towards more useful applications than any one of the component devices would be capable of supporting. DEAPspace addresses peer-to-peer networking of pervasive devices instead of client-server networking.

Example applications of DEAPspace that we envision are:

- If you receive a phone call while wearing a headset and an electronic wrist watch, your cellular phone should alert you to the call using your wrist watch, and the call should automatically be connected with your headset.
- If you are composing a GSM SMS message and a computer terminal is nearby you should be offered the option of using its keyboard instead of the cellular phone's keypad.

Our research increases the value of pervasive computing devices by allowing them to communicate via a wireless network, and share hardware resources and software services. The key physical technology that we exploit is a short-range (3 meter) wireless communication medium. In our approach, we are not limited to any particular hardware, instead DEAPspace is appropriate for any broadcast capable medium. The range of devices that we consider for DEAPspace is very broad, going from laser printers to car stereos to PDAs to wrist watch displays and to many other devices that we encounter in our environment.

2 Challenges

The challenges in achieving this scenario, broadly stated, lie with identifying the presence and capabilities of devices, and with matching complex services, such as a telephone call, with device abilities, such as voice-quality audio I/O, telephone number input, call-terminate button, etc. The former question requires both a means of specifying device capabilities, and a protocol for sharing this information with as few transmissions as possible (to preserve battery life) while at the same time provide for fast and prompt worldview updates.

Two important aspects of DEAPspace are its extensibility to future devices, and its adaptability to changing resource availability. This flexibility means that it is useful in case only a couple of devices are enabled, but increasingly so as DEAPspace becomes more pervasive—without requiring any modifications to the devices already in use.

Thus, in essence our emphasis is on transient ad-hoc, proximity-based peer-to-peer networking of PvC clients. Part of our research work is validation of our concepts and protocols through implementation: the Tuareg framework in its C and Java incarnations.

3 Status

We have developed already a very efficient and prompt service announcement and discovery algorithm that, in contrast to conventional service discovery algorithms, uses a push model instead of a pull model. Results with our RF test bed and also simulations on our DEAPspace network simulator demonstrate rather drastically that the DEAPspace service announcement and discovery algorithm performs much better than other pull-model based approaches.

We currently are preparing a version 1.0 release of the DEAPspace framework and also are working on applications of DEAPspace.

Taking DEAPspace further we want to investigate issues such as security (e.g., trust, information sharing, intrusion detection), configuration and management, and multi device applications (i.e., relayed applications, choired applications).

Generative Maßschneiderung von Betriebssystemen

Martin Becker, Lothar Baum, Lars Geyer, Georg Molter
{mbecker, lbaum, geyer, molter}@informatik.uni-kl.de
AG Systemsoftware
Universität Kaiserslautern

Die Entwicklung von Software im Bereich der eingebetteten Systeme unterscheidet sich maßgeblich von der in anderen Gebieten, da aufgrund der inhärent gegebenen Ressourcenknappheit der Einhaltung von nichtfunktionalen Anforderungen ein erheblich größerer Stellenwert zukommt. Dies äußert sich nicht zuletzt bei den eingesetzten Betriebssystemen, die häufig auf die konkreten Belange der Anwendung zugeschnitten werden müssen, um eine Realisierung des Systems bei gleichzeitiger Einhaltung strikter Vorgaben überhaupt erst zu ermöglichen. Neben den rein funktionalen Anforderungen spielen bei der Maßschneiderung insbesondere nichtfunktionale Aspekte wie Ressourcenbedarf, Fehlertoleranz und Gestaltung der Programmierschnittstelle eine maßgebliche Rolle.

Generell führen die jüngsten Bestrebungen zur effizienteren Nutzung von Wiederverwendung zu einem Richtungswechsel in der Softwareentwicklung. Neben dem Bereich der konventionellen Software wendet man sich zunehmend auch in der Domäne der eingebetteten Systeme Komponenten- und Produktlinienansätzen zu, durch die man versucht, der ingenieurmäßigen Entwicklung von Software einen Schritt näher zu kommen. Die herkömmlichen Entwicklungsmethoden müssen dabei grundlegend erneuert und auf die konsequente Identifikation, Separation und Integration von gemeinsamen und variablen Systembestandteilen ausgerichtet werden. Ein Paradigmenwechsel ist die Folge, dem sich die Betriebssystementwicklung nicht verschließen kann.

Mit der Entwicklung einer auf *Generischen Komponenten* basierenden Technologie zur teilautomatisierten Fertigung maßgeschneiderter Betriebssysteme versuchen wir diesen Rahmenbedingungen gerecht zu werden. Das Grundkonzept unseres Ansatzes bilden generische Softwarebausteine, deren Eigenschaften im begrenzten Umfang angepasst werden können, ohne dabei manuelle Veränderungen an ihrem Code vornehmen zu müssen. Die Maßschneiderung erfolgt über die Belegung von sogenannten generischen Parametern, den extern sichtbaren Stellschrauben der generischen Komponenten. Wir unterscheiden drei verschiedene Parametertypen: Während *Selektions-* und *Generierungsparameter* entsprechende Generatoren zur Durchführung der geforderten Anpassungen steuern, bieten überall dort, wo hochgradig individuelle Anpassungen erforderlich sind, *Codeparameter* den erforderlichen Freiraum für manuelle Maßschneiderung.

Der Einsatz von generischen Komponenten bei der Realisierung von Betriebssystemen ermöglicht hohe Maßschneiderbarkeit bei gleichzeitig hohem Wiederverwendungsgrad. Zusätzlich reduziert sich der Aufwand der Anpassung im Wesentlichen auf die Komponentenauswahl und die Belegung der bereitgestellten generischen Parameter. Im Falle der Selektions- und der Generierungsparameter können die implementierungstechnischen Eingriffe bei der Maßschneiderung durch die Bereitstellung entsprechender Generatoren vollständig verborgen werden – die Maßschneiderung von Betriebssystemen wird somit letztlich zur Konfigurationsaufgabe. Dieser generative Ansatz resultiert neben einer Ersparnis von Entwicklungszeit in einer Reduktion des erforderlichen Expertenwissens.

Ein inhärentes Problem von Komponentenansätzen ist dagegen die Konfigurationskomplexität. Deren Verringerung kann durch die Bereitstellung von Konfigurationswissen herbeigeführt werden. Dieses Wissen umfasst neben Einschränkungen der Kompositionsmöglichkeiten und Standardbelegungen beziehungsweise Abhängigkeiten der generischen Parameter auch Erfahrungswerte vorangegangener Maßschneiderungen und entsteht im Laufe der Entwicklung und der wiederholten Anwendung der generischen Komponenten. Zur Erfassung und Bereitstellung dieses Wissens wird in unserem Ansatz die Technik der *Erweiterten Design-Spaces* eingesetzt. Mit dieser intuitiven Methode zur semi-formalen Beschreibung von Design-Entscheidungen und –Erfahrungen lassen sich werkzeugunterstützt Komposition und Konfiguration vereinfachen; weiterhin bildet sie eine geeignete Schnittstelle für Analysewerkzeuge zur Extraktion von Betriebssystemanforderungen aus bereits existierenden Anwendungsentwürfen.

Die unterstützende Einbeziehung des Konfigurationswissens in den Maßschneiderungsprozess von Betriebssystemen führt in Verbindung mit der Technik der generischen Komponenten zu einer weitreichenden Abstraktion von den konkret erforderlichen Anpassungsschritten. Nur in Fällen, in denen noch kein passender Generator verfügbar und ein Codeparameter zu belegen ist, muss mit dem erforderlichen Expertenwissen Betriebssystemfunktionalität manuell modifiziert beziehungsweise neu erstellt werden. Über den skizzierten Ansatz kommt man dem Ziel der Maßschneiderung von Betriebssystemen direkt durch Anwendungsentwickler ein Stück näher.

Einfluß von Cache-Coloring auf die Vorhersagbarkeit und Performance von echtzeitfähigen Netzwerkkomponenten

Jork Löser

TU Dresden, Fakultät Informatik

Heutige Computersysteme setzen Speichercaches ein, um Zugriffe auf den Hauptspeicher im Mittel zu beschleunigen. Die Zeiten für Speicherzugriffe hängen dadurch im wesentlichen davon ab, ob das gesuchte Datum im Cache vorhanden ist oder nicht. Echtzeitanwendungen, die bei ihrer Ressourcenplanung vom worst case ausgehen müssen, werden so wesentlich mehr Zugriffszeit einplanen als sie im Mittel später benötigen werden.

Die Partitionierung des Cache ist eine bekannte Technik, Zugriffszeiten auf Hauptspeicher vorhersagbar zu gestalten. Echtzeitanwendungen profitieren davon, die zu planenden worst-case Zugriffszeiten werden von den wahrscheinlich auftretenden nicht mehr so stark abweichen. Darüber hinaus lassen sich einzelne Programmteile mit Hilfe der Cache-Partitionierung beschleunigen, wenn sie garantiert im Cache zu finden sind.

[1] beschreibt Cache Coloring als Technik, eine Partitionierung des Caches für Anwendungen transparent zu implementieren. Meßergebnisse weisen das prinzipielle Funktionieren nach, konkrete Anwendungen werden jedoch nicht in die Analyse einbezogen.

Anhand einer echtzeitfähigen Netzwerkkomponente wird in diesen Arbeiten der praktische Einfluß des Cache Coloring untersucht. Dazu wird ein Speichermanager benutzt, welcher Cache Coloring unterstützt. Dessen Verwendung ist für Hardwaretreiber nicht transparent, so daß der ATM-Kartentreiber und die Komponente zur Verkehrssteuerung angepaßt wurden. Die Messungen ergeben im Vergleich zum Durchschnitt teilweise doppelte worst-case Zeiten, welche durch Cache Coloring begrenzt werden können.

Performancemessungen zeigen, daß Cache Coloring Anwendungen beschleunigen kann. Die praktischen Auswirkungen sind jedoch weit weniger dramatisch, als [1] vermuten läßt. Es wird auch deutlich, daß Programme mit großen working set davon nicht profitieren. Die Meßdaten belegen Performanceeinbrüche von bis zu 80%.

[1] Liedtke, Härtig, Hohmuth: OS-Controlled Cache Predictability for Real-Time Systems. Third IEEE Real-time Technology and Applications Symposium (RTAS) 1997, Montreal, Canada.

Komponierbarkeit eingebetteter Echtzeitsysteme

Jan Richling
richling@informatik.hu-berlin.de
Humboldt-Universität zu Berlin, Institut für Informatik

13. Januar 2000

Das Design eingebetteter Systeme ist sehr eng mit der jeweiligen Anwendungsdomäne und konkreten Anforderungen der speziellen Anwendung verbunden. Aus diesem Grunde werden Konzepte wie beispielsweise Objektorientierung und Wiederverwendbarkeit, die in anderen Bereichen der Software-Entwicklung Stand der Forschung sind, nur sehr wenig eingesetzt. Eingeschränkte Ressourcen und insbesondere oft vorhandene strenge zeitliche Anforderungen eingebetteter Systeme erschweren den Einsatz solcher Technologien.

Auf der anderen Seite sind die Entwicklungskosten bei eingebetteten Systemen vor allem dann sehr hoch, wenn zeitliche Anforderungen garantiert werden müssen und solche Garantien beispielsweise gegenüber unabhängigen Zertifizierungsinstanzen bewiesen werden müssen. An dieser Stelle ist insbesondere die Benutzung von „Standardbausteinen“, sogenannten Komponenten, und deren Wiederverwendung von großer Wichtigkeit und praktischer Relevanz, da der Aufwand für Test, Verifizierung und Zertifizierung erheblich gesenkt werden kann.

Ein aktuelles Forschungsgebiet, das sich mit der Übertragung solcher Konzepte aus der Welt der offenen Systeme auf verteilte (und eingebettete) Echtzeitsysteme beschäftigt, ist die Untersuchung der „Komponierbarkeit“ solcher Systeme und die Entwicklung von Architekturen, die „Komponierbarkeit“ unterstützen. Das bedeutet an dieser Stelle, daß in einer solchen Architektur Eigenschaften, die auf der Ebene einer Komponente gegeben sind (beispielsweise ein garantiertes zeitliches Verhalten), durch die Systemkomposition nicht verändert werden und im System ebenfalls gültig sind.

Gegenstand dieser Arbeit ist die Diskussion von Ansätzen zu solchen Architekturen, wobei im Schwerpunkt *Message Scheduled System* (MSS) [2] und *Time Triggered Architecture* (TTA) [1] stehen. MSS ist eine am Institut für Informatik der Humboldt-Universität Berlin entwickelte Architektur, die davon ausgeht, daß die zeitlichen Aspekte des Systems nicht im Detail bekannt sind, sondern daß im voraus Ober- bzw. Untergrenzen bekannt sind — beispielsweise der minimale zeitliche Abstand zwischen zwei Ereignissen eines Typs, aber nicht die exakten Zeitpunkte solcher Ereignisse. MSS faßt Angaben dieser Art zu Parametern wie Auslastung von Knoten oder Kommunikationssystemen zusammen, und stellt Möglichkeiten vor, auf dieser Basis Komponenten zusammenzufügen, ohne bestehende zeitliche Garantien zu verletzen. Diese Parameter werden in Komponentenbeschreibungen zusammengefaßt, die Entscheidungen zur Komponierbarkeit unterstützen. Die Komposition selbst — also das Hinzufügen oder Entfernen von Komponenten — ist zur Laufzeit des Systems auf der Grundlage solcher Beschreibungen möglich, ohne gültige zeitliche Garantien zu verletzen. MSS ist ein betriebssystemnahes System, das Einfluß auf das Scheduling sowohl auf Prozessorebene, als auch auf Netzwerkebene hat, um diese Art der Komponierbarkeit zu realisieren.

TTA ist eine an der TU Wien entwickelte zeitgesteuerte Architektur, die es mit Hilfe des exakten Vorauswissens über alle zeitlichen Aspekte des Systems ermöglicht, in der Planung freigelassene Zeitfenster auszunutzen und Komponenten so zusammenzusetzen, daß die zeitlichen Interaktionsmuster zusammenpassen. Sämtliche Schnittstellen sind im Zeitbereich exakt definiert und verändern sich damit bei der Systemintegration nicht, womit die Architektur komponierbar in Bezug auf das zeitliche Verhalten ist. Die exakte Kenntnis des zeitlichen Ablaufes ist allerdings auch ein Nachteil von TTA, der mit einer geringen Flexibilität erkauft wird.

Abgeschlossen wird die Arbeit mit vergleichenden Betrachtungen beider Ansätze, Konzepten zur Komponetenbeschreibung, Überlegungen zur Implementation von MSS auf Basis von LynxOS oder PURE, und der Vorstellung von Simulationsergebnissen.

Literatur

[1] Hermann Kopetz. *Real-Time Systems — Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, 1997.

[2] Jan Richling. Komponierbare Echtzeitsysteme — Entwurfsmethode und Architekturentwurf. Technical Report Informatik Bericht 127, Institut für Informatik, Humboldt-Universität, Berlin, Germany, Sep 1999.

Komponentenbasierte Softwareentwicklung für eingebettete Echtzeitsysteme

Uwe Rastofer

Universität Erlangen-Nürnberg
Lehrstuhl für Betriebssysteme

Januar 2000

Viele Komponentenmodelle, wie Microsoft COM und Java Beans, basieren darauf, daß eine uniforme Ausführungsplattform und nahezu unbegrenzte Ressourcen zur Verfügung stehen. Es wird angenommen, daß sich die Komponenten nur über ihre (funktionalen) Schnittstellen beeinflussen. Nichtfunktionale Eigenschaften werden nicht betrachtet. Ein Komponentenmodell für eingebettete Echtzeitsysteme muß jedoch nichtfunktionale Eigenschaften, wie Laufzeiten und Speicherverbrauch, mit berücksichtigen, da diese die Korrektheit des Systems wesentlich bestimmen (siehe harte Echtzeitsysteme). Die nichtfunktionalen Eigenschaften der einzelnen Komponenten und des Gesamtsystems sind jedoch von der Ausführungsplattform abhängig, was zu einer geringen Wiederverwendbarkeit und Anpaßbarkeit führt.

Hier wird ein Komponentenmodell vorgestellt, das die Funktionalität der Komponenten von nichtfunktionalen Aspekten trennt (siehe Aspect-Oriented Programming). Die Komponenten sind ereignisgesteuert und ihr Verhalten wird zunächst unabhängig von einer Ausführungsplattform von Programmierer beschrieben. Aus diesen plattformunabhängigen Komponenten wird dann die Anwendung aufgebaut. Der Anwendungsentwickler kann nun zusätzliche Anforderungen an das Verhalten des Gesamtsystems in Form von Zeitbedingungen an den Eingangs- und Ausgangsereignissen definieren. Die Verbindung zwischen zusammengehörenden Eingangs- und Ausgangsereignissen erfolgt durch kritische Pfade, die durch das Gesamtsystem gelegt werden.

Erst nachdem die Funktionalität und die Zeitbedingungen des Gesamtsystems feststehen, wird die Ausführungsplattform festgelegt. Mit Hilfe einer Beschreibung der Plattform oder durch Messungen können Laufzeiten und Speicherverbrauch bestimmt werden. Anhand des daraus abgeleiteten Ablaufplanes werden Komponenten identifiziert, die nebenläufig ausgeführt werden und daher koordiniert werden müssen. Diese Komponenten werden mit Synchronisationscode ausgestattet und der Ablaufplan wird entsprechend angepaßt. An dem so entstandenen System lassen sich nun die Zeitbedingungen der Anwendung überprüfen.

Das hier vorgestellte Modell erlaubt eine einfache Wiederverwendung der Komponenten und eine Anpassung an verschiedene Ausführungsplattformen unter Beachtung nichtfunktionaler Eigenschaften. Aufgrund der Kommunikation zwischen den Komponenten über Ereignisse ist auch eine Erweiterung für verteilte Anwendungen einfach möglich.

Systemerstellung unter dem Echtzeitbetriebssystem Ambrosia/MP

Dipl.Inform. Ingo Stierand

Universität Oldenburg
Abteilung Betriebssystem und Verteilte Systeme

Januar 2000

Im Vergleich zu klassischen Rechensystemen hat im Bereich der eingebetteten Systeme die möglichst optimale Nutzung der vorhandenen Hardware-Ressourcen einen hohen Stellenwert. Neben dem Kostenfaktor spielen hier oft räumliche Beschränkungen eine Rolle. In größeren Systemen, wie sie beispielsweise im Automobilbereich vorzufinden sind, zeigt sich ein eindeutiger Trend zu vernetzten und verteilten Rechensystemen. In dem Projekt Ambrosia/MP wird das gleichnamige Echtzeitbetriebssystem entwickelt, mit dem eine effiziente Ressourcennutzung in verteilten, eingebetteten Systemen möglich ist.

Ambrosia/MP zeichnet sich durch die Fähigkeit aus, ausgehend von einem Basissystem anwendungsangepasste Laufzeitsysteme zur Verfügung zu stellen. Dabei werden bereits zur Entwicklungszeit mit Hilfe von sogenannten Komponenten die verwendeten Systemdienste vollständig an die Hardware und an die Anforderungen der Anwendung angepasst. Die entstehenden Laufzeitsysteme enthalten dann genau die für die Ausführung relevanten Elemente. Die Auswahl und Instanziierung der Komponenten basiert zum Einen auf der zugrundeliegenden Rechnerstruktur und zum Anderen auf den durch den Anwendungsentwickler mittels entsprechender Werkzeuge eingegebenen Konfigurationsinformationen.

Dieses Komponentenmodell lässt eine hohe Komplexität bei der Erstellung der Laufzeitsysteme, insbesondere in Systemen mit mehreren Recheneinheiten, erkennen. Die Umsetzung der Struktur- und Konfigurationsinformationen in ein lauffähiges System erfordert ein komplexes Regelwerk, das innerhalb eines Übersetzungswerkzeuges kodiert sein muss. An dieser Stelle setzt die im Rahmen des Projektes entwickelte Konfigurationssprache *AMScribe* an. Das dabei umgesetzte Sprachkonzept verfolgt mehrere Ziele. Um zu vermeiden, dass bei Änderungen bzw. Erweiterungen der Betriebssystembasis immer auch die Werkzeuge anzupassen sind, kann das gesamte zur Umsetzung notwendige Regelwerk innerhalb der Sprache ausgedrückt werden. Ein generischer, als Bibliotheksmodul in das Werkzeug integrierter Übersetzer wird mit dem Regelwerk instanziiert und übernimmt dann die Systemerzeugung. Um die Interoperabilität zu anderen Entwicklungswerkzeugen, wie beispielsweise Case-Tools oder Spezifikationswerkzeugen zu vereinfachen, werden alle zur Systemerstellung notwendigen Parameter ebenfalls in der Sprache ausgedrückt, so dass ein einfacher Im- bzw. Export von Konfigurationsinformationen möglich ist. Desweiteren unterstützt das Sprachkonzept die grafische Darstellung und Manipulation der Konfigurationsparameter. Als Bibliotheksmodul organisiert stehen entsprechende Dialogfelder zur Verfügung, mit denen die Entwicklung von Konfigurationswerkzeugen wesentlich vereinfacht wird.

PROTOKOLLE ZUR INTERNEN UHRENSYNCHRONISATION IN SICHERHEITSKRITISCHEN ECHTZEITSYSTEMEN

Horst F. Wedde, Wolfgang Freund
Informatik III
Universität Dortmund
44221 Dortmund

ZUSAMMENFASSUNG

Um verteilte Realzeitexperimente ausführen zu können, insbesondere für sicherheitskritische Anwendungen, wie sie in unserem MELODY Projekt untersucht werden, muß man eine Uhrensynchronisation mit einer Genauigkeit von ca. 100 μsec zwischen einer besonderen Uhr (hier *master* genannt) und irgendeiner anderen Uhr des Systems (hier *slave* genannt) gewährleisten.

Mit der Benutzung der "probabilistischen Methode zum Lesen einer entfernten Uhr", die unbegrenzte Kommunikationsverzögerungszeiten berücksichtigt, stellen wir drei neue verteilte Uhrensynchronisationsprotokolle vor. Es wird die "*masking technique* " verwendet, durch die Nachrichten mit "ungeeigneten" Verzögerungszeiten keine Wirkung haben.

Das erste Protokoll, das als *Real-Time Network Protocol (RTNP)* bezeichnet wird, basiert auf *unicast roundtrip* Nachrichten und wurde auf RS/6000 Maschinen unter AIX in einem *Token-Ring* Netzwerk implementiert. In RTNP synchronisieren die *slaves* ihre lokale Uhr, indem sie eine Nachricht zu dem *master* schicken. Gleich nachdem der *master* diese Nachricht empfangen hat, schickt er eine Rückmeldung mit seiner aktuellen Zeit. Wenn dann diese Nachricht beim *slave* innerhalb eines off-line berechneten Zeitfensters ankommt, kann er seine lokale Uhr verstellen, sonst muß er diesen Vorgang wiederholen.

Bei nochmaliger Untersuchung von RTNP in unserem neuen experimentellen Netzwerk, das aus sieben *LINUX-PC*s mit einem 100 Mbps *Ethernet* besteht, ergab sich, dass dieses Protokoll nicht harmonisch war, das heißt, dass die Abweichung zwischen zwei *slaves* deutlich größer ist als die Abweichung zwischen dem *master* und einem *slave*. (Das ist bei allen auf *unicast* basierten Protokollen zu erwarten.) Nach ausführlicher Untersuchung dieses Phänomens entwarfen wir zwei neue Protokolle: Das *Real-Time Duplex Protocol (RTDP)* und das *Real-Time Burst Protocol (RTBP)*, die beide auf *broadcast* Nachrichten basieren. Im Gegensatz zu einer Sequenz von *unicast* Nachrichten werden *broadcast* Nachrichten im Idealfall von allen Knoten gleichzeitig empfangen. Durch die Verwendung angemessener *broadcast* Protokolle wird erwartet, dass geringere Uhrenunterschiede zwischen *master-slave* und *slave-slave* erreicht werden.

Obwohl RTDP und RTBP sehr verschiedene Ansätze zur Lösung des Problems benutzen, stellte es sich bei beiden heraus, dass sie bezüglich der *master-slave* Uhrenabweichung besser und stabiler als RTNP (und wahrscheinlich jedes andere auf *unicast* basierende Protokoll) sind. Das Wichtigste ist, dass wir mit diesen Verfahren eine harmonische Uhrenabweichungstruktur erreichen, in der die Abweichung zwischen zwei *slaves* dieselbe ist wie zwischen dem *master* und einem *slave*.

Außerdem ist trotz der etwas größeren Übertragungszeiten von *broadcast* Nachrichten im Vergleich zu *unicast* Nachrichten die Genauigkeit für eine *master/slave* Synchronisation besser als die *unicast* Methode.

Wir diskutieren unsere experimentellen Ergebnisse und auch wie beide *broadcast* Protokolle sowohl die Realzeitanforderungen als auch die Anforderung an die Fehlertoleranz erfüllen.

Anforderungen an die Entwicklung und Zertifizierung sicherheitsgerichteter Betriebssysteme für Embedded Systems

Rainer Faller

TÜV Product Service GmbH, München

Betriebssysteme erlauben eine wesentlich effizientere Entwicklung von komplexer Software in Embedded Systems. Bis heute können die Entwickler sicherheitsgerichteter Computersysteme die Vorteile von käuflichen (off the shelf) Betriebssystemen und Libraries jedoch nur begrenzt nutzen. Die Hersteller von Betriebssystemen hatten bisher wenig Möglichkeit und Veranlassung die Anforderungen der Funktionalen Sicherheit zu verstehen.

Diese Situation hat sich mittlerweile vielfältig geändert. Nachdem die ersten zertifizierten Betriebssysteme für sicherheitsgerichtete Embedded Systems für Avionik-Systeme und große Automatisierungssysteme verfügbar sind, wird der Wunsch nach Anwendung von Betriebssystemen auch für andere sicherheitsgerichtete Embedded Systems in KFZ-Anwendungen, Maschinensteuerungen und in der Bahntechnik immer dominanter. Parallel dazu wurde nach langen Jahren der Diskussion unter Experten mit der internationalen Norm IEC 61508-3 und der anwendungsorientierten Norm DO178B weltweit anerkannte Grundlagen für die Entwicklung sowie Verification & Validation von sicherheitsgerichteter Software verabschiedet.

Der Vortrag wird die sicherheitstechnischen Anforderungen an Software insbesondere Betriebssysteme und deren Entwicklung sowie das Zertifizierungsverfahren beschreiben. Es werden die Betriebssystemeigenschaften erläutert auf die die Entwickler sicherheitsgerichteter Softwaresysteme aufbauen können.

Gesellschaft für Informatik
Fachgruppe 3.1.4
Betriebssysteme

Dr. Jürgen Kleinöder
Universität Erlangen, Informatik 4
Martensstr. 1

91058 Erlangen

FAX: (09131) 85 - 2 87 32

Anmeldung zum Frühjahrstreffen am 02. und 03. März

Name: .....

Vorname: .....

Institution: .....

Adresse: .....

Telefon: .....

Telefax: .....

E-Mail: .....

Ankunft: .....

Abreise: .....

☐ Ich werde am 02. März um 10:47 mit dem IC 818 in Erlangen ankommen und würde den Shuttle-Dienst zum Uni-Gelände gerne nutzen[1]

☐ Ich beabsichtige an der Abendveranstaltung am 02. März (gemeinsames Abendessen) teilzunehmen (Angabe bitte für unsere Vorausplanungen - nicht verbindlich)

Datum / Unterschrift: .....

1. wir werden Sie per E-Mail oder Fax über weitere Details informieren