Forschungszentrum Informatik

an der

Universität Karlsruhe

Forschungsbereich Softwaretechnik

Dissertationsvorschlag

EventChannelNetwork – Nachrichtenkanal-Netzwerk als Basis zur Kommunikation in verteilten Anwendungen unter Echtzeitanforderungen

Marc Schanne
schanne@fzi.de

Abteilung Softwaretechnik
Forschungszentrum Informatik

Version 0.8

Contents

1 Einleitung

Nach dem erfolgreichen Einsatz verteilter Systeme in internetbasierten Geschäftssoftware-Anwendungen wird jetzt und in Zukunft auch das Feld der eingebetteten Systeme durch Verteilung von Komponenten und durch Einsatz objektorientierter Techniken unterstützt. Insbesondere eingebettete, verteilte Systeme mit Echtzeitanforderungen sind derzeit im Fokus der Forschung und Entwicklung. Auf Basis einer integrierten Entwicklungsumgebung demonstriert das von der europäischen Union geförderte Forschungsprojekt HIDOORS [1, 2, 3] die Eignung von objektorientierter Entwicklung für eingebettete Systeme. Das Forschungsprojekt AJACS [4, 5] greift insbesondere die Eignung der Java Plattform für eingebettete Systeme im Automobil unter Echtzeitbedingungen auf und mit HIJA [6] wird aktuell eine architekturneutrale Lösung für verteilte Echtzeitanwendungen, ebenfalls auf Basis von Java, erforscht und entwickelt.

Dieser Vorschlag für eine Dissertation konzentriert sich auf die notwendige Kommunikationsinfrastruktur für solche Anwendungen. Der vorgestellte Ansatz basiert auf einem auftragsorientierten Kommunikationmodell und ist so die geeignete Ergänzung zu einer direkten, synchronen Kommunikation über Aufrufe im fremden Adressbereich der beteiligen Komponenten.

2 Anforderungen

Dieses Promotionsvorhaben untersucht in wie weit der Kommunikatonsansatz mit indirekter Adressierung und einem asynchronem Nachrichtenaustausch über dezentrale Puffer sich für den Einsatz mit Echtzeitanforderungen eignet. In diesem Absatz wird deshalb kurz auf die Anforderungen an ein solches System eingegangen und eine derzeit übliche Umgebungen für ein solches System skizziert. Basierend darauf und auf einem Vergleich möglicher Kommunikationsmodelle wird die Entscheidung für den Einsatz einer Publiziere/Abonniere-Kommunikation [7] in verteilten Echtzeitanwendungen belegt.

2.1 Echtzeitanforderungen

Anforderungen, die sich auf rechtzeitige Ausführung und Beendigung innerhalb einer gegebenen Zeitgrenze beziehen lassen sich in sogenannte *harte*, *weiche* und *feste* Echtzeitanforderungen unterteilen [8]. Insbesondere durch die Auswirkungen bei Nichteinhaltung dieser Zeitanforderungen lässt sich die folgende Einteilung finden.

Harte Echtzeit Die Antwort auf eine Dienstanfrage muss innerhalb des gegebenen Zeitfensters ohne Ausfall erbracht werden. Eine Nichterbringung führt zum vollständigen Ausfall oder Fehler des Gesamtsystems, der mit ernsthaftem Schaden, evtl. auch für Leib und Leben behaftet ist.

Weiche Echtzeit Eine Anfrage an das System wird im Regelfall in der gegebenen Zeit erfüllt und eine Nichterbringung des Dienstes in der vorgegebenen Zeitgrenze hat im wesentlichen Mängel bei der Qualität des Gesamtdienstes (Quality of Service, QoS) zur Folge.

2.2 Echtzeitbussysteme

Technische Basis für verteilte Systeme ist ein zugrundeliegendes Kommunikationsnetzwerk. Im Umfeld eingebetteter Systeme haben sich unterschiedliche Netze entwickelt. Diese Feldbussysteme sind oft rundruforientiert (broadcast) und bieten ein einfaches Kommunikationsprotokoll [9]. Geringe Latenzzeiten sind neben einem deterministischen Multiplexverfahren Grundlage für die Unterstützung von Echteitanforderungen in diesem Einsatzgebiet. Echtzeitbussysteme sollen so kostenbewusst die Verbindung unterschiedlichster Hardwarekomponenten (Prozessoren, Sensoren und Aktoren) unterstützen.

Kriterium	Option	
Adressierung	direkt	indirekt
Blockierung	synchron	asynchron
Pufferung	ungepuffert	gepuffert, Mailbox
Kommunikationsform	meldungsorientiert	auftragsorientiert

Tabelle 1: Kommunikationsmodelle [10]

2.3 Integration

Die Integration dieser Kommunikationssysteme von eingebetteten Prozessoren-, Sensoren- und Aktorenbündeln in bestehende Nachrichtensysteme ist ein Ziel bei der Vereinfachung von Systemarchitekturen in Echtzeitsystemen wie z.B. im Auto mit CAN oder MOST. Mit dem Einsatz eines protokollübergreifenden Nachrichtenaustauschs innerhalb von logischen Nachrichtenkanälen soll dies mit dem hier vorgestellten Ansatz erbracht werde. Neben der Kommunikation der Komponenten untereinander ist auch die leichte Integration mit externen (evtl. Nicht-Echtzeit-) Systemen (z.B. offboard units) zur Analyse ein gewünschtes Ziel und das Nachrichtenkanal-Netzwerk muss dies durch geeignete Kommunikationsmuster bereitstellen.

3 Konzepte und verwandte Arbeiten ...

Dieser Abschnitt stellt eine Auswahl der existierenden verwandten Arbeiten im Bereich asynchroner Kommunikation für verteilte Systeme zusammen und bewertet ihre Vorteile und Einschränkungen. Ausgehend von einer Übersicht möglicher Kommunikationsmodelle und einer Begriffsbildung [10] werden die Vor- und Nachteile insbesondere asynchroner Publiziere/Abonniere-Kommunikation in verteilten Systemen beschrieben, im Unterabschnitt 3.2 werden Entwicklungen vorgestellt und abschließend ihre Eignung für die im letzten Abschnitt beschriebenen Anforderungen überprüft.

3.1 Kommunikationsmodelle

Bei der Kommunikation von verteilten Systemen auf Basis von Nachrichtenaustausch ohne gemeinsam benutzten verteilten Speicher lassen sich die in Tabelle 1 vorgestellte Kriterien zur besseren Beschreibung finden.

Bei direkter Adressierung spricht der Sender den Empfäger direkt und im allgemeinen hartcodiert an im Unterschied dazu wird bei indirekter Adressierung über eine Empfangsstelle (Mailbox) kommuniziert.

Blockierende Kommunikation führt automatisch zu einer Synchronisation der Kommunikationspartner, ein Sender wartet bis die Nachricht über das Netz losgeschickt worden ist und evtl. eine Empfangsbestätigung vom Empfänger zurückgekommen ist. Der korrespondierende Empfänger blockiert bis eine Nachricht angekommen ist.

Unabhängig von der Adressierungsart und dem Synchronisationsverhalten ist eine Pufferung auf Sender- und Empfängerseite möglich. Bei asynchroner Kommunikation weiß der Sender nicht ob und wann die Puffer frei sind. Sind die Puffer voll, so muss blockiert werden, was dem asynchronen Konzept widerspricht, oder Daten gehen verloren.

Als Kommunikationsformen lassen sich meldungsorientierte Einwegnachrichten[1] mit höchstens einer Empfangsbestätigung und auftragsorientierte Hin- und damit verbundene Rücknachrichten unterscheiden. Auch diese Formen lassen sich mit den vorgestellten anderen Kriterien kombinieren.

[1] Idealbetrachtung!

Im folgenden wird insbesondere die asynchrone Option bei der Blockierung der einzelnen Prozesse, mit indirekter Adressierung und gepuffertem Nachrichtenaustausch betrachtet. Asynchrone Kommunikation erlaubt die (zeitliche) Entkopplung von Sender und Empfänger, Parallelarbeit wird unterstützt.

Asynchrones Verhalten ist auf den 1. Blick zwar ein Widerspruch zu Echtzeit, die eine synchrone und somit deterministische Ausführung erfordert, aber durch die Bindung der asynchronen Ausführungszeiten an Zeitgrenzen und vorherberechenbare Ausführungszeiten unter ungünstigsten Voraussetzungen lässt sich diese Kommunikation synchronisieren.

Die folgenden Abschnitte stellen das Publiziere/Abonniere-Muster für asynchrone Kommunikation, sowie die Vor- und Nachteile asynchroner Kommunikation unter Echtzeitbedingungen detaillierter dar und sind Grundlage für den Vergleich von existierenden Techniken und Produkten in Abschnitt 3.2.

3.1.1 Publiziere/Abonniere-Kommunikation

Grundsätzlich lassen sich zwei Verfahren bei Publiziere/Abonniere-Kommunikation unterscheiden. [7] stellt themenbasierte (Topic-based) und inhaltbasierte (Content-based) Protokolle gegenüber. Die inhaltbasierten Protokolle erweitern die themenbasierte Abonnierung von Nachrichten um eine auf den aktuellen Inhalt der Nachrichten bezogene Selektion. Die Empfänger von Nachrichten beschreiben einen Filter und abhängig davon wird die Auswahl getroffen.

Themenbasierte Protokolle nutzen im Gegensatz dazu vorher festgelegte Ausdrücke (Themen) und der Empfang von Nachrichten eines Themas impliziert die Gruppierung der Kommunikatonspartner. Durch eine hierarchische Strukturierung der Themen und implizite Abonnierung von Unterthemen lässt sich diese flache Kommunikation erweitern. Der hier vorgestellte Ansatz der Kommunikation im Nachrichtenkanal-Netzwerk bietet in der 1. Ausbaustufe nur eine einfache Thema-zu-Echtzeitanforderung-Zuordnung im Nachrichtenkanal, evtl. Erweiterungen hierzu sind zwar denkbar aber zur besseren Erfüllung der Anforderung rechtzeitiger und schneller Ausführung wird erstmal darauf verzichtet. Weitere Einzelheiten zum Entwurf der Kommunikationsinfrastruktur bietet der Abschnitt 4.

3.1.2 Vor- und Nachteile asynchroner Kommunikation im Echtzeiteinsatz

Obwohl eine asynchrone und zeitentkoppelte Kommunikation im 1. Moment den deterministischen und statisch verifizierbaren Anforderungen in sicherheitskritischen Echtzeitsystemen widerspricht, bietet die Publiziere/Abonniere-Kommunikation verschiedene Vorteile für verteilte Systeme und die Anpassung an Echtzeitsysteme versprechen lohnenswerte Ziele:

Anonyme Kommunikation: Sender benötigen keine Information über die Adressen und die Anzahl der Empfänger und genauso müssen die Empfänger nicht die Identität des Senders kennen;

Entkopplung von Publizierer und Abonnent verspricht größere Flexibilität und eine einfachere Wiederverwendung von Programmcode durch die Entkopplung in Adressierung, Zeit und Synchronisation;

Viele-zu-viele-Kommunikation erlaubt die Ausnutzung von rundruforientierten Feldbussystemen im Umfeld von Eingebetteten Systemen;

Skalibarkeit: Einfache erweiterbare asynchrone Kommunikation ohne blockierende Anfrage/Antwort Interaktionen skaliert gut von einfachen bis großen Systemen;

Einfache Programmierung von Steuerungsgeräten in der Automation: Verwandtes Kommunikationsmodell: Informationen müssen nicht gelesen werden, kein Wissen über interne Speicherabbildungen, oder Datenbankstrukturen, oder unbekannter Steuerungsgeräte ist notwendig;

Effiziente Nutzung von Netzwerkbandbreite: Kein Netzwerkverkehr für Anforderungen, Nutzung direkter ereignisgesteuerter Kommunikation;

Robustes Applikationsdesign: Unterstützung für Ausfallsicherheit und Migration im Fehlerfall;

Portabilität über Plattformgrenzen: Die Kommunikationsinfrastruktur ist zwar als Implementierung auf Basis der Echtzeit-Java-Spezifikaton (RTSJ) ausgelegt, das Kommunikationsprotokoll selbst ist aber von Systemarchitektur, Programmierplattform und Netzwerktechnik unabhängig;

Ausrichtung auf Echtzeitanforderungen: Geeignet im Einsatz mit Signalströmen, Statusaktualisierungen, ereignisgesteuerten Befehlen in Echtzeitsystemen.

Um diese Vorteile asynchroner Kommunikation für das Anwendungsgebiet deterministischer, echtzeitfähiger Systeme zu erschließen müssen vorhersagbare Grenzen auf Sender- und Empfängerseite in die Kommunikation eingebaut werden. Die Garantien, die die hier vorgestellte Nachrichtenkanalkommunikation bietet, werden im Abschnitt 4 theoretisch und im Unterabschnitt Ablaufkoordinierung mit ein paar praktischen Rahmenvorgaben vorgestellt und diskutiert.

Die in den folgenden Unterabschnitten vorgestellten Produkte, Forschungsprojekte und Programmierschnittstellen bieten einen Überblick über aktuelle konkurrierende Entwicklungen im Umfeld verteilter Kommunikation. Sie sind in 1. Linie anhand ihrer Bereitstellung von asynchronen Kommunikationscharakteristika für unterschiedliche Anwendungsgebiete zusammengestellt. Insbesondere ihre Eignung für den Einsatz in echtzeitkritischen Systemen wird abschließend im Unterabschnitt 3.3 zusammengefasst, verglichen und bewertet.

3.2 Produkte, Forschungsprojekte und Programmierschnittstellen

Im Umfeld verteilter Systeme sind recht unterschiedliche Kommunikationsinfrastrukturen entwickelt worden. Die folgenden Unterabschnitte stellen eine Auswahl kurz vor und in Abschnitt 3.3 wird ihre Eignung bezüglich der Echtzeitanforderungen zusammengefasst.

3.2.1 Synchrone Anfrage/Antwort Architektur

In Java wird bei synchroner Kommunikation mit entfernten Methodenaufrufen seit der Version 1.1 der Einsatz von RMI (Remote Method Invocation [11]) angeboten. RT-RMI (Real-Time RMI) für echtzeitfähige Systementwicklung ist derzeit Teil der Forschung [12]. RMI konkurriert direkt mit dem plattformübergreifenden Kommunikationsmodell von CORBA (Common Object Request Broker Architecture) auf dessen Basis die Kommunikaton zwischen Anwendungen verschiedener Sprachen und Plattformen ermöglicht wird.

3.2.2 Echtzeit CORBA mit Nachrichten- und Benachrichtigungsdienst

Mit der Einführung von Echtzeitanforderungen im CORBA Softwareentwurf (RT-CORBA [13]) hat die OMG (Obejct Management Group) zwei interessante asynchrone Kommunikationsprotokolle auf Basis der etablierten synchronen Infrastruktur definiert. Auf Basis der synchronen Kommunikation kann mit RT-CORBA beziehungsweise mit unterschiedlichen ORB Implementierungen (Beispiel: TAO [14]) so auf einen Nachrichten- und Benachrichtungsdienst [15, 16] zugegriffen werden.

3.2.3 Java Messaging Service (JMS)

Echte asynchrone Kommunikation im Java Umfeld wurde 2002 mit der Definition von JMS (Java Messaging Service [17]) eingeführt. Dieser skalierbare, asynchrone Nachrichtendienst bietet sowohl themenbasierte Publiziere/Abonniere-Kommunikation als auch inhaltsbezogene Filterung der übertragenen Nachrichten an. Die Architektur selbst ist zentralisiert.

3.2.4 Jini, JavaSpaces, JXTA

Für die Implementierung von dezentralen Ad-Hoc-Netzwerken zwischen unterschiedlichsten Internet-fähigen Endgeräten hat Sun Microsystems ein auf RMI basierendes Protokoll für Java-Anwendungen: Jini [18]. Durch Einsatz von Zwischenspeichern (JavaSpaces [19]) analog zu einer schwarzen Tafel können asynchrone Kommunikationsstrukturen implementiert werden. Anwendungen, die eine Gleiche-zu-gleichen-Kommunikationform (Peer-to-Peer, P2P) verwenden lassen sich so auf Basis von Eins-zu-eins-Kommunikation mit entfernten Methodenaufrufen entwickeln.

Ausgehend von dienstorientierter Anwendungeentwicklung hat ein breites Konsortium von Soft- und Hardwareanbietern auf Basis von Java die API Open Software Gateway Initiative (OSGi) definiert. Dieser ebenfalls auf Ad-Hoc-Netzwerke ausgerichteter Ansatz für die Verteilung von Diensten basiert zwar ebenfalls auf einem Netzwerkansatz für die Verteilung und das Auffinden von Diensten, arbeitet aber ansonsten in einem lokalen Service-Container, der für den Zugriff auf Software und Hardwaredienste gedacht ist.

Eine leichtgewichtigere und weniger auf Java ausgerichtete Variante[2] für die Entwicklung von P2P Anwendungen ist seit 2001 die JXTA Plattform [20]. Basierend auf 6 zugrundliegenden Protokollen (Suche, ???) ermöglicht dieses System die Entwicklung von verteilten Gleiche-zu-gleichen-Anwendungen. Die Kommunikation selbst ist immer synchron und asynchrones Verhalten wird allein durch lokale Pufferung von Anfragen und Anforderungen ermöglicht. Im Abschnitt 6.5 wird die Integration und Verbindung der JXTA Protokolle mit der Infrastruktur und den Kommunikationmöglichkeiten des vorgestellten Nachrichtenkanal-Netzwerks weiter betrachtet und als mögliche Erweiterung diskutiert.

3.2.5 Java InfoBus

Zur lockeren Kopplung von Java Komponenten (JavaBeans) innerhalb einer virtuellen Maschine hat Sun Microsystems ein eng mit diesem Komponentenkonzept verwobenes Protokoll zur Interaktion definiert. Der Java InfoBus [21] definiert einen Themen-bezogenen Informationsbus, mit dem die direkte Kommunikation über Methodenaufrufe ergänzt werden kann. Dieser Ansatz wird derzeit nicht in Hinblick auf Verteilung über mehrere virtuelle Maschinen und Adressbereiche ausgedehnt und es sind keinerlei Aussagen über Einhaltung von Zeitbedingungen bei der Interaktion vorgesehen.

3.2.6 JavaGroups

JavaGroups [22, 23] definieren ebenfalls ein gruppenbasiertes und anonymes Kommunikationsprotokoll. Dieses Open-Source Projekt wird z.B. bei der netzwerkweiten Kopplung von Enterprise-Java-Containern des JBoss Projektes zur Bildung von Rechnerbündeln (cluster) eingesetzt.

3.3 ... und ihr Eignung für Echtzeit-Systeme

Die Kommunikation im Bereich der eingebetteten Systeme erfordert eine besonders auf die Leistungsbegrenzungen und eingeschränkte Kommunikationskapazität ausgerichtete Kommunikationsinfrastrukturen. Gegen zentralisierte Architekturen sprechen Faktoren wie Zuverlässigkeit und Ausfalltoleranz, sowie das Problem von zentralen Fehlerpunkten, die die Funktionstüchtigkeit gesamter Anwendungen beeinträchtigen können.

Die Eignung der vorgestellten Systeme in Hinblick auf Echtzeitfähigkeiten ist nur für die Echtzeiterweiterungen der CORBA Infrastruktur definiert.

???

[2] Es gibt bereits Implementierungen auf Basis von C oder Perl

4 Entwurf und Implementierung: EventChannelNetwork

Der verwendete Ansatz einer direkten Publiziere/Abonniere-Kommunikation nutzt die technischen Rahmenbedingungen der möglichen Bussysteme im Bereich engebetteter Systeme. [24] stellt für die 1. Implementierung im Rahmen des HIDOORS Forschungsprojektes drei unterschiedliche Bussysteme (CAN, TTP/C, Ethernet) und die notwendige Fehlerbehandlung gegenüber.

In HIDOORS wurden die Anforderungen an den Nachrichtendienst in der 1. Version wie folgt definiert:

- Benachrichtigung zwischen Prozessen in verschiedenen Adressbereichen verschiedener Prozessoren
- Nachrichtenkommunikation mit geringer Latenz
- Nachrichten werden direkt aufgenommen und zur Verarbeitung geeigneten Echtzeitkontrollfäden übergeben, die strikte Zeitanforderungen garantieren
- Dienst entkoppelt Sender und Empfänger für flexible und einfache Wiederverwendung
- Verschiedene Komponenten der Applikation kommunizieren über einen logische Nachrichtenkanal, dies ermöglicht die gemeinsame Nutzung einer technischen Kommunikationsinfrastruktur
- Verwaltung des Dienstes ist verteilt und besitzt keinen einzelnen Punkt für Fehler
- Implementierung des Prototypen erfolgt in Echtzeit-Java
- Kommunikationsprotokoll abstrahiert von unterschiedlichen technischen Netzwerken. Der zusätzlich notwendige Protokollaufwand ist gering und der Nachrichtendienst nutzt die gebotenen Protokoll und Pakete sowie Netzwerkkommunikatonsstruktur um die im technischen Netzwerk gegebenen Garantieen an die Komponenten des Nachrichtendienstes weiterzureichen.

Um den so entstandenen Dienst auch für den Einsatz von sicherheitskritischen, eingebetteten Systemen mit harten Echtzeitanforderungen ausbauen zu können verfolgt die 2. Version des Nachrichtdienstes insbesondere das Ziel der

- Vorhersagbarkeit.

Der Prototyp des Nachrichtenkanal-Netzwerks baut deshalb auf Programmierbibliotheken und Umgebungen der Java 2 Plattform mit der Echtzeiterweiterung RTSJ (Real-Time Specification for Java [25]) auf.

Um sowohl für Anwendungsfälle mit harten Echtzeitbedingungen statisch verifizierbare Interaktionen, als auch für 'nur' geschäftsprozesskritische Anwendungen mit weichen Echtzeitbedingungen ein vollständiges dynamisches Kommunikationsmodell bereitstellen zu können basiert die Implementierung des im folgenden näher beschriebenen Nachrichtendienstes auf zwei verschiedenen Profilen. Im Kontext des Forschungsprojektes HIJA (vgl. Abschnitt 4.5.2) werden diesem im Folgenden als HRTJ (Hard Real-Time Java) und SRTJ (Soft Real-Time Java) Profile klassifiziert. Die gleichen Konzepte des Nachrichtenkanal-Netzwerkes für HRTS Systeme werden als Spezialisierung der SRTJ Variante mit einem dynamischen Verhalten gesehen (vgl. Abbildung 1). Die SRTJ-Implementierung verwendet normale Java-Ausnahmen um Fehlverhalten bei der Kommunikation zu signalisieren und die Applikationsschicht ist für eine geeignete Behandlung verantwortlich [24]. Die Spezialisierung für HRTJ bietet im wesentlichen eine direkte Vererbung der Klassenstrukturen und Schrittstellen. Methoden, die im dynamischen all eine Ausnahme werfen können sind hier aber gegen Methoden ohne Ausnahmebehandlung ausgetauscht worden. Evtl. Fehler sollten in der Entwicklungsphase statisch verifiziert werden können und trotzdem auftretende Unregelmäßigkeiten bewirken nicht behandelbare Programmfehler.

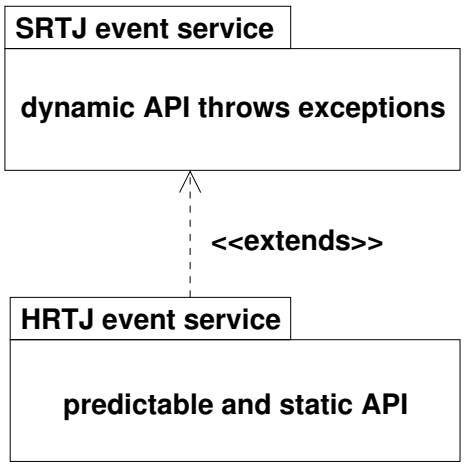

Figure 1: Profile des HIJA Nachrichtendienstes

4.1 Java, RTSJ

Die Entwicklung der Java Plattform hin zur Unterstützung von eingebetteten Systemen ist durch Definition von unterschiedlichen Editionen in der Java 2 Plattform weiter verbessert worden. Zusätzlich dazu erlauben weitergehende Bibliotheken und Definitionen mit Erweiterungen für die virtuelle Maschine der Java Laufzeitumgebung auch die Nutzung im Umfeld hater Echtzeitanforderungen.

Das gesamte Feld eingebetteter und auch mobiler Systeme wird durch die Java 2 Micro Edition unterstützt. Die Palette dieser unterschiedlichen, anwendungsorientierten und reaktiven kleinen Systeme wird durch die Partitionierung in zwei Konfigurationen und eine weitere Differenzierung mit Profilen, in denen Bibliotheken definiert werden unterstützt.

Mit Definition einer Echtzeit Spezifikation für Java (Real-Time Specification for Java, RTSJ) ist es gelungen die Semantik der Java-Sprache um nforderungen in harten und weichen Echtzeitumgebungen zu erweitern. Mit der RTSJ wird sowohl eine spezielle virtuelle Maschine, als auch eine umfassende Bibliothek für die Bereitstellung von Echtzeit-Ablaufkoordinierung, Speichermanagement, Hardwarezugriff und Synchronisation definiert.

Der hier vorgestellte Entwurf asynchronen Kommunikationsmusters im Nachrichtenkanal-Netzwerk basiert auf diesen beiden Entwicklungen in der Java-Plattform und zeigt insbesondere auch die Notwendigkeit für Weiterentwicklungen in diesem Feld, aber das eigentliche Kommunikationsprotokoll ist plattformneutral und kann auch auf Basis anderer Echtzeitumgebungen implementiert werden.

4.2 Komponenten

Für das bessere Verständnis der Struktur und Interaktion im Nachrichtenkanal-Netzwerk werden in diesem Abschnitt sämtliche Komponenten und ihr Zusammenspiel zur Erbringung der asynchronen und deterministischen Kommunikation beschrieben. Ausgehend vom Nachrichtenkanal selbst werden die auf ihm interagierenden aktiven Objekte sowie notwendige Kommunikationswarteschlangen definiert [26].

Abschnitt 4.2.7 illustriert diese Struktur in einer Zusammenfassung und die Verbindung zu existierenden Entwurfsmustern wird hier gezogen und diskutiert.

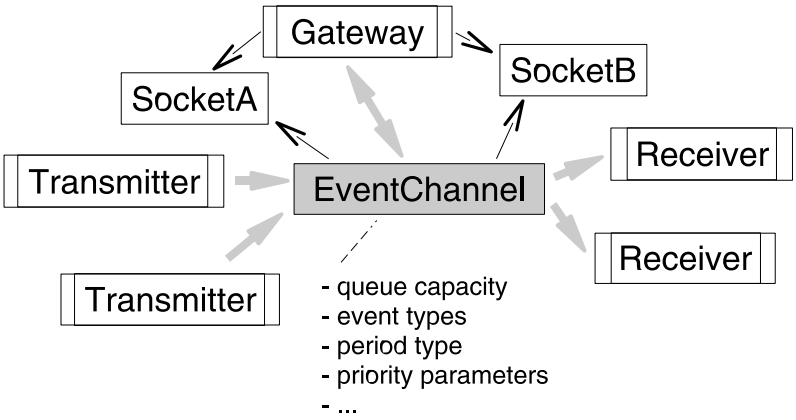

Figure 2: Nachrichtenkanal

4.2.1 Nachrichtenkanal

Der Nachrichtenkanal (EventChannel, vgl. Abbildung 2) ist die zentrale Komponente im vorgestellten Nachrichtendienst. Der Nachrichtenkanal repräsentiert das Thema der Publiziere/Abonnenten-Kommunikation und hier werden neben den möglichen Nachrichtentypen über diesem Kanal auch Anforderungen an die aktiven Komponenten, die Nachrichten senden und empfangen definiert. Ihre Aufgaben werden im Folgenden beschrieben.

4.2.2 Empfänger-Kollektiv

Das Empfänger-Kollektiv umfasst die aktiven Komponenten eines Knotens, die für das Empfangen von Nachrichten zuständig sind. Um den fehlerfreien Empfang aller Nachrichten zu gewährleisten müssen diese Kontrollfäden mit höchster Priorität[3] arbeiten und nach dem Empfang einer Nachricht wird diese zur weiteren Verarbeitung in eine nach Prioritäten sortierte Warteschlange (vgl. Abschnitt 4.2.6) eingereiht. Eine genauere Betrachtung der Prioritäten aller aktiven Komponenten im Nachrichtenkanal-Netzwerk wird in Abschnitt 4.3 bereitgestellt.

4.2.3 Zugriffssockel

Der Zugriff auf das dem Nachrichtenkanal-Netzwerk zugrunde liegenden physikalischen Netzwerks wird über einfache Byte-orientierte Zugriffssockel garantiert. Diese Sockel bieten den Lese/Schreibe-Zugriff auf das Kommunikationsnetzwerk, das im wesentlichen Funktionalität vergleichbar der ISO/OSI Schicht 5 (Transportschicht) anbieten muss. Hier wird kein zuverlässiges Protokoll als Basis vorausgesetzt. Der Zugriffssockel ermöglicht ohne weiteren Protokolloverhead die Utialisierung auch von z.B. Echtzeitgarantien. Für einen Vergleich möglicher Kommunikationsnetzwerke und -protokolle vergleiche Abschnitt 4.4. Es ist denkbar, dass der Zugriffssockel neben der direkten Interaktion mit dem zugrundeliegenden Netzwerk weitere Funktionalität für die Aufteilung und Verschmelzung von Nachrichtenpaketen in Netzwerktransporteinheiten, sowie die in Abschnitt 4.2.2 angeregte Bandierung und Oberbegrenzung von Prioritäten der empfangenden Kontrollfäden enthalten kann. Voraussetzung für den Einsatz mit Echtzeitnetzwerken muss dieser zusätzliche Protokollaufwand begrenzt und nach oben abschätzbar bleiben.

[3] Um zu verhindern, dass niederprioritäre Nachrichten allein durch ihren Empfang die Bearbeitung wichtigere Nachrichten behindern wird hier ein Verfahren mit Prioritätsbändern [13] für die verwendeten Zugriffssockel eingesetzt und so die Partitionierung der Rechenkapazität abhängig von Prioritäten ermöglicht.

4.2.4 Weitere aktive Komponenten für die Kommunikation

Neben einem Empfängsthread wird auf den Zugriffssockel auch durch eine Komponente für die Versendung (Transmitter) und evtl. die Netzwerkeüberbrückung (Gateway) zugegriffen. Analog zum Empfangsthread werden die Parameter dieser aktiven Komponenten durch den Nachrichtenkanal vorgegeben. Diese Komponente ist für den asynchronen Versand der Nachrichten und die Interaktion mit dem zugrundeliegenden Kommunikationsnetzwerks notwendig. Der Versand der Nachrichten kann gepuffert sein um unterschiediche Zeitfenster eines z.B. zeitfensterbasierten Protokolls zu synchronisieren. Die erforderliche Priorität und Parameter für die Periode, sowie verfügbare Warteschlangenkapazitäten sind hier erforderlich.

Eine denkbare Netzwerkbrücke zwischen unterschiedlichen zugrunde liegenden Kommunikationsnetzwerken vereinigt sowohl Empfänger- als auch Versenderfunktionalität und unterliegt dabei den Prioritätsanforderungen des Nachrichtenkanals.

4.2.5 Aktivitätsmanager und Pool von Aktivitäten

Der Aktivitätsmanager ist als aktive Komponente für die Abarbeitung der Warteschlangen des Empfänger-Kollektivs verantwortlich und garantiert die Zuordung wartender, gelesener Nachrichten zu freien Aktivitätskontrollfaden aus einem Pool. Wartende Kontrollfäden mit ausrichender Priortät werden aktiviert und die Abarbeitung registrierter Bearbeitungslogik (PushEventHandler) für die Nachrichten mit ihnen gestartet.

4.2.6 FIFO-Warteschlangen

Die Interaktion aller aktiven Komponenten wird durch eine Kopplung mit FIFO-Warteschlangen unterstützt. Sowohl zwischen Empfänger-Kollektiv und Aktivitätsmanager, als auch bei der Zuteilung von Nachrichten an Aktivitäten wird dieses Verfahren eingesetzt.

4.2.7 Zusammenfassung

Die soweit eingeführten Komponenten können in einem neuen Architekturentwurfsmuster zusammengefasst werden. Durch die Interaktion eines Kollektivs von Empfängern, der asynchronen Interaktion mit einer Managerkomponente, die empfangene Nachrichten anhand ihrer Priorität durch einen Pool verfügbarer Aktivitäten ausführen lässt, wird auf Basis einfacher synchroner Kommunikationsnetzwerke und -hardware ein asynchrones Kommunikationsmuster mit deterministischen Zeitbegrenzungen erlaubt.

Entwurfsmuster: Empfänger-Kollektiv, mit Aktivitätsmanager und Warteschlagen [26] Die Abbildung 3 illustriert die Struktur der hier vorgestellten Komponenten im Nachrichtenkanal-Netzwerk. Das eingeführte Entwurfsmuster erfordert keine speziellen asynchronen Netzwerkzugriffs- oder Betriebssystemunterstützung. Der Entwurf ist ähnlich des Reaktormusters [27] und basiert ebenfalls auf einem synchronen Nachrichten-Demultiplexer, erweitert diesen aber um den Aktivitätsmanager und eine asynchrone Warteschlangeninteraktion.

Verwandte Entwurfsmuster Das im letzten Absatz beschriebene Entwurfsmuster basiert im Wesentlichen auf der Integration schon bekannter Muster.

Reaktor [28]:

Beobachter [29]:

Aktivator:

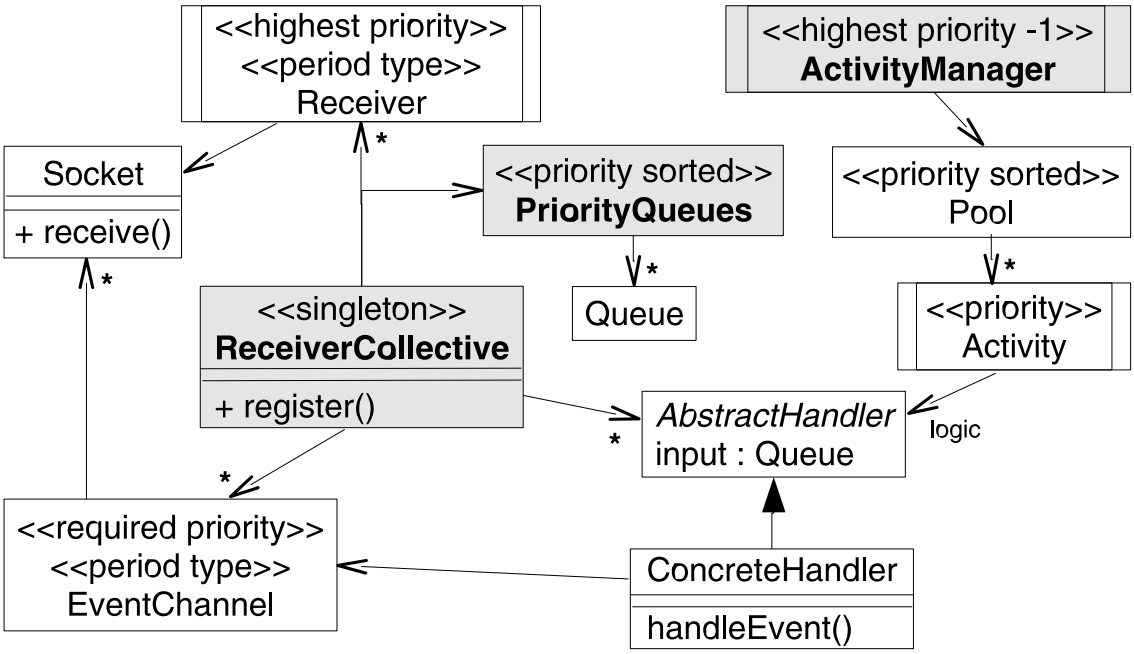

Figure 3: Struktur Nachrichtenkanal-Netzwerk

4.3 Ablaufkoordinierung (Scheduling)

Die in den Abschnitten 4.2.2, 4.2.4 und 4.2.5 vorgestellten aktiven Komponenten des Nachrichtenkanal-Netzwerks erfordern eine geordnete Ablaufkoordinierung um eine Verarbeitung der Nachrichten in Echtzeit mit Einhaltung angegebener Zeitfristen zu garantieren.

Diese Ablaufkoordinierung lässt sich innerhalb einer virtuellen Maschine für die Kommunikation eines Knotens im Nachrichtkanal-Netzwerk statisch berechnen und verifizieren. Im HRTJ Profil werden Echtzeitkontrollfäden mit periodischer oder sporadischer Ausführung definiert [30]. Diese beiden Formen werden auch auf Nachrichtenfrequenzen der definierten Nachrichtenkanäle abgebildet. Durch Einsatz einer festen, prioritätsgebundenen Ablaufkoordinierung mit einem Laufzeitsystem, das ein Priority Ceiling Emulation Protokoll implementiert lassen sich mit Techniken der Rate Monotonic Analysis (RMA) [8] die Abwesenheit von Verklemmungen (deadlocks) statisch analysieren und nachweisen. Für Systeme mit harten

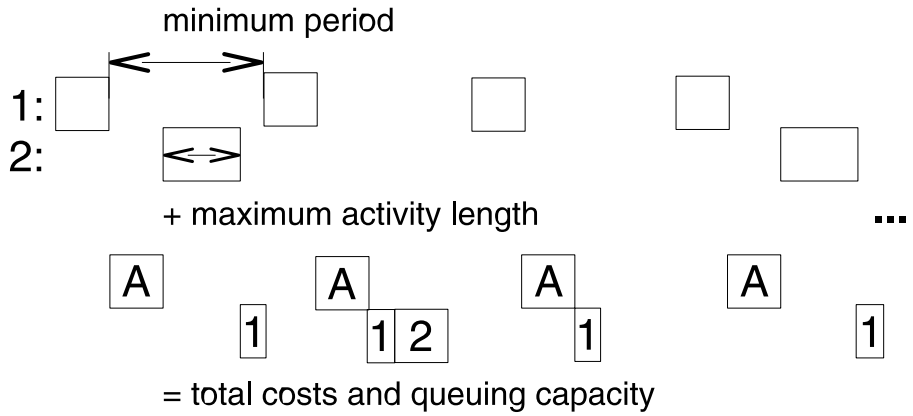

Abbildung 4: Beispiel der Ablaufkoordinierung mit zwei Empfängerkontrollfäden

Echtzeitanforderungen wird in HIJA die Segmentierung der Anwendung in zwei Phasen (Initalisierungs- und Missionsphase) propagiert [31]. Auch diese Aufteilung nutzt des Nachrichtenkanal-Netzwerk und verlagert die nicht zeiiitkritischen Aktionen der Initalisierung in eine gesonderte Phase vor der eigentlichen echtzeitigen Ausführung.

Das in Abbildung 4 dargestellte UML-Zeitdiagramm beschreibt die eigentliche Kommunikation in der Missionsphase und lässt sich auf seine Ausführbarkeit hin prüfen.

Für den Einsatz mit SRTJ-Systemen bietet die Implementierung des Nachrichtenkanal-Netzwerkes nicht diese Garantien. Mit dem leichtgewichtigen und dezentralen Kommunikatonsprotokolls wird aber ein möglichst hoher Grad an Servicequalität (Quality of Service, QoS) angestrebt.

4.4 Netzwerke

???

4.4.1 CAN, Realtime Ethernet mit UDP

Das Controller Area Network (CAN [32]) wurde von der Firma Bosch für die Kommunikation in Industrieanlagen und elektronischen Sterungsanlagen entwickelt und findet heute verbreitet im Automobil Einsatz. Durch sein nachrichtenbasiertes Kommunikationsmodell unterstützt CAN aber von sich aus nur weiche Echtzeitanforderungen. Erst durch Verbindung der Nachrichtenkommunikation mit Vergabe der Identifizierer der Nachrichten, die direkt auf die Priorität der Übertragung Auswirkungen haben, kann das Protokoll auch für harte Echtzeitanforderungen genutzt werden [???].

4.4.2 Time Triggered Architecture und TTP/C

Durch ein zeitbasiertes Multiplexverfahren und die Ressourcenzuteilung bietet eine zeitgesteuerte Architektur (Time Triggered Architecture [33]) Garantien, die für die Bereitstellung von harten Echtzeitanforderungen notwendig sind. ???

4.4.3 AFDX

Für den Einsatz im Luftfahrtbereich entwickelte Kommunikationsnetze und -protokolle basieren ebenfalls auf statisch berechenbaren, zeitfensterbasierten Verfahren. Das Avionics Full Duplex Switched Ethernet (AFDX/ARINC 664 [34]) Protokoll ist ???

4.5 Projekte

Der erfolgreiche Einsatz des EventChannelNetworks wurde und wird derzeit in zwei Forschungsprojekten, die von der europäischen Kommission gefördert sind, belegt. ???

4.5.1 HIDOORS

Das von der Europäischen Union geförderte Forschungsprojekt HIDOORS (High Integrity Distributed Object-Oriented Realtime Systems) untersuchte die Voraussetzungen für eine integrierte Entwicklungsumgebung für eingebettete Systeme basierend auf der Java 2 Plattform. Mit der Bibliothek für Nachrichtenkommunikation wurden verteilte Anwendungen auf Basis des EventChannelNetworks in UML entworfen und durch automatisierte Quellcode-Generierung große Teile der notwendigen Kommunikationsinfrastruktur ohne Mehraufwand erzeugt. Die Eignung des Nachrichtenkanal-Netzwerkes für verteilte, eingebettete systeme wurde insbesondere mit einem Demonstrator aus dem Avionic-Umfeld belegt. Weitere Informationen zur Testumgebung und -ergebnissen bietet der Abschnitt 5.1.1.

4.5.2 HIJA

Mit High Integrity Java Applications (HIJA) wird in einem weiteren EU-geförderten Forschungsprojekt die Eignung von Java bei der Anwendungsentwicklung im Umfeld Eingebetteter Systeme mit Echtzeitanforderungen

5 Bewertung

Für die Bewertung des vorgestellen Ansatzes einer asynchronen Publiziere/Abonniere-Kommunikation bieten die in Abschnitt 4.5 vorgestellten Forschungsprojekte verschiedene erfolgreiche Testszenarien und Beispielapplikationen.

5.1 Demonstrator-Projekte

Die Entwicklung für das Forschungsprojekt HIDOORS ist eine Vorläuferimplementierung des in Abschnitt 4 vorgestellten Entwurfsmusters. Die grundlegenden Komponenten dieses Kommunikationsrahmenwerks waren zwar schon mit der endgültigen Implementierung im Projekt HIJA vergleichbar, aber insbesondere die Aufteilung in zwei Profile für weiche und harte Echtzeitanforderungen war noch nicht vorgesehen.

Im Forschungsprojekt HIJA gibt es sowohl Beispielszenarien mit harten (vgl. HIJA - Avionics), als auch mit weichen oder gemischten Echtzeitanforderungen (vgl. HIJA - AI).

5.1.1 HIDOORS - Avionics

SkySoft, Datagram, dynamische Erzeugung von Kommunikationskanälen ???

5.1.2 HIJA - AI

Telecom Italia, SRTJ, HeyWoW ???

5.1.3 HIJA - Avionics

Thales, HRTJ, statische Analyse ???

6 Erweiterungen

???

6.1 Deklarative Beschreibung mit XML

Die im Abschnitt 4 vorgestellte Struktur des Nachrichtenkanal-Netzwerkes eigenet sich neben der statischen Verfikation der aktiven Komponenten auch für eine einfache statische und deklarative Beschreibung auf Basis von XML.

Als Nebenergebnis innerhalb des HIDOORS Forschungsprojekts (vgl. Abschnitt 4.5.1) ist diese als Vorläufer der im folgenden Abschnitt beschriebenden Generierung aus UML Diagrammen entwickelt worden.

6.2 Generierung aus UML

Ausgehend vom Ansatz einer deklarativen Beschreibung des Kommunikationsverhaltens mit XML wurde insbesondere für das Forschungsprojekt HIDOORS (vgl. Abschnitt 4.5.1) die Deklaration von Laufzeitparametern und Interaktionsschemata innerhalb von Interaktions- und Klassenstrukturdiagrammen der UML mit der Erweiterung für Ablaufkoordinierung und Zeitbeschreibung [???] implementiert.

6.3 Integration mit OSGi

Durch die Implementierung der Kommunikationsfunktionalität als OSGi Service ??? Einsatz von Partitionierungstechniken bei den aktiven Komponenten des Nachrichtendienstes ??? Unterstützung für Programmcodeaustausch ???

6.4 Verifikation durch Modellprüfung

Das in Abschnitt 4.3 vorgestellte Verfahren der Ablaufkoordinierung für harte Echtzeitanforderungen innerhalb einer virtuellen Maschine durch zusätzliche Protokollbeschreibungen bei der Kommunikation zwischen unterschiedlichen Adressbereichen um eine weitere statische Verifikaton erweitert werden. Im Projekt HIDOORS (vgl. Abschnitt 4.5.1) wurden diese Informatonen af Basis der UML-Diagramme im Entwurf der verteilten Anwendung annotiert. HIJA (vgl. Abschnitt 4.5.2) ermittelt hingegen auf Basis von Strukturen und Annotatonen im Quellcode der Anwendung selbst ein vereinfachtes Kommunikationmodell. Dieses Zustandsmodell wird zur Verifikation mit mit einem echtzeitfähigem Modellprüfer, z.B. UPPAAL [???], herangezogen und der Progammierer beim Entwurf der Anwendung durch eine weitere Zeitanalyse für ungünstigste Fälle (Worst Case Execution Time Analysis, WCETA) unterstützt.

6.5 P2P für Eingebettete Systeme mit dem Nachrichtenkanal-Netzwerk

Die dezentrale Struktur des Nachrichtenkanal-Netzwerkes legt die Nutzung im Zusammenhang mit fehlertoleranten und eingebetteten Geiche-zu-Gleichen-Kommunikatonssystemen (Peer-to-Peer, P2P) nahe. Durch die leichte Skalierbarkeit asynchroner Kommunikation wird hier die Entwicklung von P2P Anwendungen für Eingebetteter Systeme ermöglicht. Als Programmierschnittstelle ist hier z.B. eine Implementierung für die JXTA Protokolle [20] denkbar.

Eine Alternative zu der von Sun Microsystems initiierten Programmierbibliothek für Gleich-zu-gleiche-Kommunikation wird im Demonstratorprojekt 'Ambient Intelligence' (vgl. Abschnitt 5.1.2) des HIJA Projektpartners Telecom Italia auf die frei verfügbare Bibliothek FreePastry aufgesetzt. Sie implementiert P2P Kommunikation nach Pastry und wurde an der Rice University entwickelt. Durch die Adaption des Nachrichtenkanal-Netzwerkes um die jeweilige Schnittstellen und Interaktionen von Pastry wird die Basis für einen proritären fehlertoleranten, asynchronen Kommunikationsdienst ermöglicht. ???

7 Zusammenfassung

In diesem Dissertationsvorschlag wird auf Basis einer direkten Publiziere/Abonniere-Kommunikation mit dezentralen Warteschlangen das asynchrone Kommunikationsmuster für echtzeitkritische Systeme geöffnet.

Literatur

[1] "HIDOORS - High Integrity Distributed Object-Oriented Realtime Systems. Project Website." http://www.hidoors.org, 2002.

[2] J. Ventura, F. Siebert, A. Walter, and J. J. Hunt, "HIDOORS - A high integrity distributed deterministic Java environment," in *Proceedings of the The Seventh IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002)*, 2002.

[3] "HIDOORS Methodology Handbook," 2003.

[4] "AJACS - Applying Java to Automotive Control Systems. Project Website." http://www.ajacs.org, 1999.

[5] "AJACS - Applying Java to automotive Control Systems. Final Report," July 2002. IST-1999-12504.

[6] "HIJA - High Integrity Java Applications. Project Website." http://www.hija.info, 2004.

[7] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, pp. 114–131, June 2003.

[8] M. H. K. ant Thomas Ralya, B. Pollak, R. Obenza, and M. G. Harbour, *A Practitioner's Handbook for Real-Time Analysis. Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.

[9] K. Etschberger, ed., *CAN - Controller Area Network. Grundlagen, Protokolle, Bausteine, Anwendungen*. Carl Hanser Verlag, 1994.

[10] P. D. R. Seck, *Skripte und Umdrucke zur Vorlesung Verteilte Systeme 2004*, ch. Kommunikation. http://cd1.ee.fhm.edu, 2004.

[11] Sun Microsystems, *Java Remote Method Invocation (RMI) Specification*, 1999.

[12] A. Wellings, R. Clark, D. Jensen, and D. Wells, "A Framework for Integrating the Real-Time Specification for Java and Java's Remote Method Invocation," in *Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp. 13–22, 2002.

[13] Object Management Group, Inc., *Real-Time CORBA Specification*, version 1.1, formal/02-08-02 ed., August 2002.

[14] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design of the TAO Real-Time Object Request Broker," *Computer Communications, Elsivier Science*, vol. 21, April 1998.

[15] Object Management Group, Inc., *CORBA Event Service Specification*, 1.1 ed., March 2001.

[16] Object Management Group, Inc., *CORBA Messaging*, May 1998.

[17] Sun Microsystems, *Java Message Service*, 1.1 ed., April 2002.

[18] Sun Microsystems, *Jini Specification*, 2.0 ed., June 2003.

[19] Sun Microsystems, *JavaSpaces Service Specification*, 2.0 ed., June 2003.

[20] *JXTA ???*

[21] Sun Microsystems, *InfoBus 1.2 Specificaton*, 1.2 ed., February 2004.

[22] B. Ban, "Adding Group Communication to Java in a Non-Intrusive Way Using the Ensemble Toolkit," tech. rep., Dept. of Computer Science, Cornell University, November 1997.

[23] B. Ban, "JavaGroups - Group Communication Patterns in Java," tech. rep., Dept. of Computer Science, Cornell University, July 1998.

[24] M. Schanne and D. J. J. Hunt, "Remote Event Service Design," tech. rep., FZI Forschungszentrum Informatik, 2004. Deliverable D4.2 describing the HIDOORS event channel network.

[25] G. Bollella, B. Brosgol, P. Dibble, S. Furr, J. Gosling, D. Hardin, M. Turnbull, and R. Belliardi, *The Real-Time Specification for Java*, 1.0 ed., June 2000.

[26] M. Schanne, "Real-Time Communication with a Receiver Collective, Activity Manager, and Queues," in *Proceedings of IADIS Internatonal Conference Applied Computing 2005*, 2005.

[27] D. C. Schmidt, "Reactor. An Object Behavioral Pattern for Demultiplexing and Dispatching Handles for Synchronous Events," in *Proceedings of the First Pattern Languages of Programs Conference*, 1994.

[28] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-oriented Software Architecture. Patterns for Concurrent and Networked Objects*, vol. 2. John Wiley & Sons Ltd., April 2001.

[29] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Entwurfsmuster*. Addison-Wesley, 1996.

[30] A. J. Wellings, "Computational Models to Support Timing Analysis," tech. rep., The University of York with input from all partners, January 2005. Deliverable D1.2a describing the HIJA Computational Models.

[31] P. Puschner and A. J. Wellings, "A Profile for High-Integrity Real-Time Java Programs," in *Proceedings of the 4th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, 2001.

[32] R. B. GmbH, *CAN Specification*. Robert Bosch GmbH, Postfach 50, D-7000 Stuttgart 1, 2.0 ed., September 1991.

[33] TTA-Group, Schoenbrunner Strasse 7, A-1040 Vienna, Austira, *Time-Triggered Protocol TTP/C. High-Level Specification Document*, 1.0.0 ed., July 2002.

[34] Condor Engineering, Inc., Santa Barbara, CA 93101, *AFDX / ARINC 664 Tutorial (1500-049)*, 1.0 ed., November 2004.