Smart Objects – Auswirkungen massengedruckter Einfachelektronik auf die IT-Infrastrukturen

Smart Objects – Effects of Mass-printed Electronics on IT Infrastructures

Michael Beigl, TU Braunschweig,
Till Riedel, Christian Decker, Universität Karlsruhe

Zusammenfassung Neue Entwicklungen in der Elektronik wie massengedruckte Einfachelektronik ermöglichen den Einsatz von Computersystemen an oder gar in Objekten, für die bisher ein solcher Einsatz nicht wirtschaftlich sinnvoll war. Dieser Beitrag untersucht die prinzipiellen Rahmenbedingungen bei der nahtlosen Integration von Smart Objects in bestehende IT-Strukturen. Probleme bei der Einbindung solcher Smart Objects in IT-Systeme – insbesondere hinsichtlich der Skalierbarkeit – werde thematisiert. Er zeigt zudem innovative Lösungsansätze wie „Active Interfaces" und Push der Geschäftslogik in Smart Objects. Der vorgestellte Systemansatz profitiert dabei von einer Vielzahl preiswerter, zusammenarbeitender Smart Objects und kann somit als Blaupause für die Informationsverarbeitung und Integration künftiger massengedruckter Elektroniksysteme in IT-Systeme gesehen werden. ►►►

Summary New developments in electronics as simple mass printed electronics enable the implementation of computing systems at or even in objects – an application that was not economically nor technical feasible before. This paper investigates principle conditions when integrating Smart Objects seamlessly into existing IT infrastructures. We also report on problems when connecting Smart Objects to IT-Systems. Based on an implemented system, we show innovative solutions to approach this problem, namely the use of active interfaces and push of business logic to Smart Objects. The approach uses multiple Smart Objects that collaborate to perform a task. It can therefore be seen as a blue print of integrating future tiny computing systems based on printed electronics into IT infrastructures.

KEYWORDS C.5 [Computer Systems Organization; Special Purpose and Application-Based Systems]; J.7 [Computer Applications: Computers in other Systems]; smart objects, wireless sensor systems, business integration, IT infrastructure / kabellose Sensorknoten, Integration von Geschäftsprozessen, IT-Infrastruktur

1 Überblick

Die Integration massengedruckter Einfachelektronik wird es ermöglichen, IT-Systemen eine hohe Anzahl von Informationen bereitzustellen. Dies können einfache Informationen wie Identifikatoren von Objekten sein. In Zukunft werden auch aus Sensordaten gewonnene Informationen bereitgestellt werden, z. B. Informationen über den Zustand einer Ware bei Transport oder Lagerung. Sensorbasierte Informationen werden zum Teil heute schon erfasst, allerdings zumeist manuell oder durch Unterstützung einfacher elektronischer Systeme, z. B. mittels RFID Technologie. Systeme, die eine Überwachung pro Gut sowie zusätzliche Sensorik vorsehen, befinden sich in der Erprobung. Die Erfassung erfolgt hier zeitlich gestreckt; eine durchgehende Echtzeitfähigkeit der Systeme ist nicht möglich.

Mit Hilfe des Massendrucks besteht die Möglichkeit, in jedes Objekt – z. B. in jeden Artikel – Elektronik zu integrieren, und dort einfache Funktionsabläufe eigenständig durchführen zu können. Mit Hilfe dieser Technologie werden also Objekte zu smarten Objekten (*Smart Objects*). Der bisherige Ansatz zur Integration solcher Smart Objects in existierende IT-Infrastrukturen ist die direkte, nicht vorverarbeitete

Einspeisung der anfallenden Information in IT Backend-Systeme als Ersatz für die bisherige manuelle Eingabe von Informationen. Allerdings erweist sich der erhebliche Umfang der eingehenden Information als problematisch hinsichtlich ihrer Skalierbarkeit und Komplexität. Wir vertreten in diesem Artikel die These, dass die Integration von Smart Objects einen vollständig anderen Ansatz benötigt: Anstatt Smart Objects an IT-Infrastruktur anzukoppeln, sollte *IT-Anwendungslogik in Smart Objects eingebunden werden*. Da solche smarten Objekte wenig Rechenleistung besitzen, ist es notwendig, die Durchführung der Aufgabe kollaborativ in Zusammenarbeit zwischen den Smart Objects zu bewältigen.

Die aufgestellte These stellt einen fundamentalen Paradigmenwechsel für IT-Systeme dar. Wir werden in diesem Artikel anhand eines realen Beispiels Implementierung und Auswirkungen aufzeigen und zwei Kerntechniken für die Einbindung von Smart Objects in IT-Infrastrukturen, nämlich Active Interfaces und Logic Push, vorstellen.

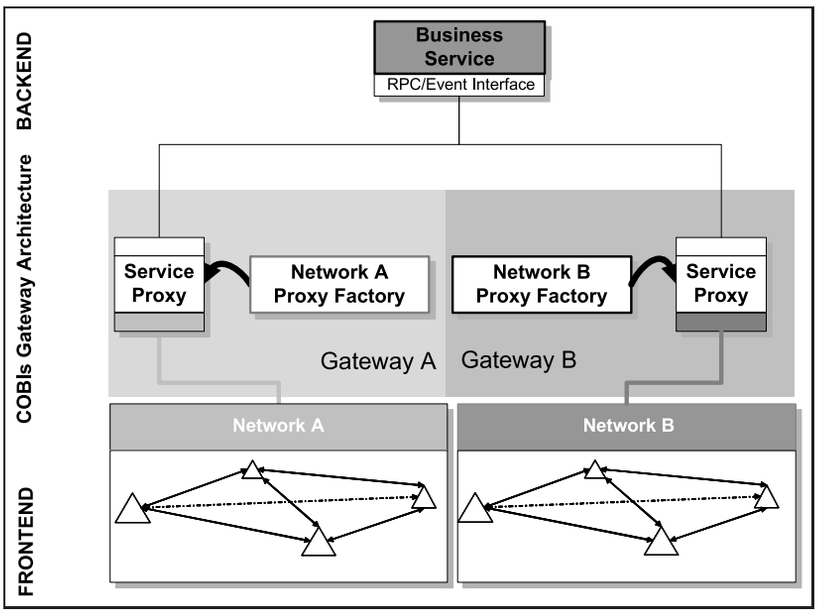

Bild 1 Smart Objekts Integration und Generic Interfaces über Service Proxies.

Beispiel: Einsatz von Smart Objects in einer Chemiefabrik

Im folgenden Beispielszenario wurde eine Versuchsinstallation von Smart Objects in einer Chemieanlage bei BP in Hull, UK, durchgeführt und an die dort vorhandene IT-Infrastruktur angebunden. Dabei wurden Container mit Chemikalien durch Einbettung von Elektronik zu Smart Objects erweitert. Als Technologie wurden kabellose Sensorknoten gewählt (Bild 1). Die Elektronik enthielt Sensorik, Funkschnittstelle und einfache Informationsverarbeitung sowie eine kleine Batterie. Obwohl diese Technologie derzeit für einen breiteren Einsatzbereich noch nicht kostengünstig genug ist, ist durch den Einsatz von gedruckter Massenelektronik eine entsprechende Preisreduktion und damit eine Ausweitung des Einsatzbereichs erwartbar. Spezifisches Anwendungsziel der Versuchsanordnung war die nahtlose Überwachung der Lagerbedingung für alle chemischen Güter, und die automatische Echtzeit-Benachrichtigung bei fehlerhafter Behandlung (z. B. beim Transport) oder bei nicht spezifikationsgerechten Lagerbedingungen.

Die in dem Chemielager zu verwendenden Regularien waren bisher schon innerhalb der IT-Infrastruktur als Teil eines Workflow-Systems hinterlegt. Dort werden die genauen Bedingungen für Behandlung und Lagerung der Güter festgeschrieben. Diese Regularien werden derzeit durch die Mitarbeiter manuell überwacht bzw. eingehalten. Für die Integration von Smart Objects in die IT-Infrastruktur ergaben sich zwei Optionen: Die Anbindung der Smart Objects als reine Sensordatenlieferanten, oder die Einbindung der Smart Objects in den Ablauf der Geschäftsanwendung zur Überwachung der Lagerabläufe.

Nach Analyse des Anwendungsumfelds wurde sich für die zweite Durchführungsoption entschieden. Die Hauptgründe dafür waren:

- Die Anbindung von Smart Objects als reine Sensorknoten würde eine permanente Kommunikation zwischen Smart Objects and IT-Infrastruktur erfordern. Dies konnte im weitläufigen Bereich der Chemieanlage nicht gewährleistet werden.
- Die Warnung für das Personal aufgrund von falschen Behandlungen musste in Echtzeit durchgeführt werden. Diese Durchführung ist bei Verwendung traditioneller IT-Systeme nicht möglich, da hier Latenzzeiten von vielen Sekunden bis Minuten möglich sind.
- Die Anzahl der zu übermittelnden Rohdaten ist sehr groß und verursacht eine hohe Kommunikations- und Rechenlast und entsprechende Kosten im Backendsystem, sowie hohe Energiekosten in den Sensorsystemen.

In unserem Feldversuch kommunizieren die Smart Objects lokal untereinander und überwachen in Kollaboration die Umgebung um die Container herum, sowie die Lagerbedingungen des Containers selbst. Dabei werden auch komplexe Überwachungsbedingungen realisiert, z. B.

- die kollaborative Überwachung der Anzahl der Container,
- Nähe der einzelnen sich in den Container befindlichen Chemikalien zueinander,

- korrekte Lagerung von Chemikalien an dafür vorgeschriebenen Orten und deren Inventur,
- singuläre Parameter wie Erschütterung und Temperatur.

Die kollaborative Überwachung findet mit Hilfe von *collaborative reasoning* [1] Technologien statt. Die speziellen Regeln für die Überwachung werden direkt von der IT-Infrastruktur bezogen und auf die Smart Objects aufgespielt (Logic Push, siehe unten). Da die Änderungsrate für die Regeln gering ist, spielen Verbindungsausfälle sowie Bandbreite des Sensornetzwerks zu IT-Infrastruktur eine untergeordnete Rolle.

Im Erkennungsfall können durch eingebaute visuelle Signale, Übertragung an ein persönliches Gerät einer in der Nähe befindlichen Person, sowie durch Übermittlung an die IT-Infrastruktur Warnungen ausgelöst werden. Die echtzeitfähige Warnmöglichkeit vor Ort stellt eine schnelle Behandlung der Situation sicher. Eine eventuell verzögerte Warnung an die IT-Infrastruktur dient vor allem der Bekanntmachung der Situation für die Werksüberwachung sowie zum Logging der Situation. Die technische Anbindung zwischen den nativen Schnittstellen der Sensornetzwerke und der IT-Infrastruktur erfolgt über Gateways (siehe Bild 1). Diese bilden eine Zwischenschicht zwischen Sensornetzwerken und IT-System und ermöglichen so einen zweistufigen Transformationsprozess von Informationen.

Das hier beschriebene System der nahtlosen Integration von Smart Objects und IT-Infrastrukturen erfordert ein neuartiges technisches Rahmenwerk. Im Folgenden werden wir dieses Rahmenwerk vorstellen, und mit Hilfe der beschriebenen Anwendung veranschaulichen.

2 Analyse

Eine typische Geschäftsanwendung (engl.: *business process*) wird durch Transformation von Ressourcen beschrieben, die dazu geeignet sind, einen messbaren geschäftsrelevanten Fortschritt zu erzielen. Geschäftsprozesse können in Aufgaben (*process tasks*) aufgeteilt werden, die dann auf verschiedene Dienstgeber (engl.: *service provider*) verteilt werden. Die Sequenz der Aufgaben kann in der sogenannten Geschäftslogik (engl.: *business logic*) erfasst und modelliert werden.

Der Begriff Geschäftslogik wird benutzt, um die Verarbeitung von Informationen von deren Präsentation und Speicherung innerhalb der klassischen 3-tier IT-Architektur unterscheiden zu können. Die Geschäftslogik befindet sich dabei in der mittleren Stufe und stellt somit die Verbindung zwischen Präsentation und Speicherung dar. Die Datenspeicherung wird meist CRUD (*Create, Read, Update and Delete*) genannt und referenziert damit auf die vereinfachten Basisprimitive, die von dieser Stufe angeboten werden. Da diese Primitive sehr einfach gehalten sind, beinhalten viele IT-Systeme noch ein erhebliches Wissen über den verwendeten CRUD, was die Wiederverwendbarkeit oder den Austausch dieser Schicht bzw. die Anbindung eines anderen CRUD-Systems erschwert.

Service-orientierte Architekturen können verwendet werden, um die enge Kopplung aufzulösen und Schnittstellen über alle beteiligten Systeme hinweg ohne die Notwendigkeit der Einordnung ihrer Funktion anzubieten. Unsere Sichtweise ist es, dass ein Service eine wohldefinierte Funktionalität und eine dazugehörige wohldefinierte Schnittstelle enthält, die einen Zugriff auf die Funktionalität gewährt. Dazu soll die Dienstabstraktion so gestaltet sein, dass die Gestaltung eines Services unabhängig von der Gestaltung der anderen Services durchgeführt werden kann. Diese lose Kopplung unterscheidet die *Service Oriented Architekture* von anderen Architekturen und erst dadurch wird eine flexible Einbindung neuer Technologie wie Smart Objects möglich.

2.1 Geschäftslogik und Smart Objects

Geschäftslogik kann als virtuelle Repräsentation der Abläufe in der physikalischen Welt aufgefasst werden. Jedes Objekt (z. B. eine Ware) im Ablauf eines realen Geschäftsprozesses besitzt damit ein Äquivalent innerhalb des IT-Systems. Geschäftsregeln und Arbeitsflüsse geben an, wie mit dem Objekt zu verfahren ist.

Kollaborativ zusammenarbeitende Smart Objects verlagern die bisherige Schnittstelle zum IT-System weg vom menschlichen Benutzer oder automatisierten Lesegerät hin zum Smart Object selbst und damit hin zur realen Welt und deren Ereignissen. Dies ermöglicht es IT-Systemen und Geschäftsprozessen, nicht mehr nur auf Bedingungen der Welt (zeitverzögert) zu reagieren, sondern mit Vorgängen innerhalb der physikalischen Abläufe zu interagieren. In unserem Beispiel dient die informationstechnische Repräsentation von Smart Objects als Statthalter der physikalisch existierenden Chemie-Container, und im IT Sinne sind sie direkt als Service Provider ansprechbar. Smart Objects fügen damit einem Objekt die folgenden Fähigkeiten hinzu (siehe auch [3]):

- Rechenfähigkeit,
- Datenspeicherung,
- Beobachtung (*Monitoring*),
- Überwachung (*Controlling*),
- Kommunikation.

Derzeit werden für die Realisierung von Smart Objects kabellose Sensorknoten wie z. B. Berkeley Motes [6], Ambient uNodes [7] oder Particle Computers [8] eingesetzt.

2.2 Implementierung von IT-Logik auf Smart Objects

Die von einem einzelnen Smart Object zur Verfügung stehenden Rechenressourcen sind im Vergleich zu anderen IT-Systemen sehr eingeschränkt. Microcontroller mit geringer Bus-Breite (8-bit oder kleiner) und wenigen Kilobyte oder gar wenigen Byte Speicher sind hier im

Einsatz. Um die Aufgaben dennoch bewältigen zu können, werden Aufgaben von Smart Objects über das Netzwerk gemeinschaftlich gelöst. So können auch einfache Systeme komplexen Lösungen überlegen sein, was an zwei Beispielen verdeutlicht werden soll:

Das erste Beispiel ist die Verteilbarkeit von Geschäftslogik. Während in einem zentralen System ein System für alle Objekte die Berechnung durchführen muss, kann im Falle der verteilten Geschäftslogik dieses auf den Smart Objects selbst geschehen. Die Verteilung der Logik kann dabei dynamisch anhand der zu leistenden Aufgaben und der verfügbaren Smart Objects erfolgen. Dadurch lassen sich z. B. Kommunikationsflüsse und die Netzwerklast optimieren. Beispielsweise müssen Informationen nicht bzw. nicht über weite Strecken zur Auswertung übertragen werden, sondern werden zunächst lokal verarbeitet. Kommuniziert werden dann abstrakte Ergebnisse dieser lokalen Prozesse. Durch Vorverarbeitung und die lokal beschränkte Kommunikation werden Latenzzeiten verringert. Das Smart Object basierte IT-System ist damit auch skalierbar: Neue Aufgaben sind in der Regel mit der Einbringung neuer (smart) Objects verbunden, die dann für zusätzliche Verarbeitungsaufgaben genutzt werden können.

Das zweite Beispiel bezieht sich auf die Lokalität typischer Überwachungsaufgaben, wie sie für einen Großteil der Geschäftslogik typisch ist. In unserem Beispiel ist dies der Abgleich von gewünschten Lagerbedingungen von Objekten zueinander. Die Nutzung der Lokalität ist bei Implementierung von Geschäftslogik auf Smart Objects systembedingt implizit gegeben, müsste bei Implementierung auf einem zentralen System aber nachberechnet werden.

2.3 Existierende Forschungsarbeiten

Die Mehrzahl existierender Service-orientierter Konzepte zur Einbindung von Smart Objects konzentriert sich auf die Darstellung von Smart Objects als Datenquellen (siehe z. B. [2]). Andere Arbeiten erlauben die Implementierung statischer, einfacher Logik innerhalb der Smart Objects, die dann über eine IT-Infrastruktur über Datenbank-ähnlichen Anfragesprachen abgefragt werden können (siehe z. B. [10]). Wieder andere Konzepte erfordern den Umbau existierender IT-Infrastrukturen, um Smart Objects integrierten zu können (siehe z. B. [9]).

Im Gegensatz zu den erwähnten Arbeiten soll das hier vorgestellte Konzept die nahtlose Integration von Smart Objects in existierende IT-Infrastrukturen zulassen. Dies wird dadurch ermöglicht, dass Smart Objects als Dienstgeber transparent in IT Struktur eingebettet werden können.

3 Entwurf

Die Integration von Smart Objects in eine IT-Infrastruktur erfordert die Analyse und eventuelle Transformation der verwendeten Architekturstile und deren Koordinationsmechanismen sowie der Kommunikationsmechanismen. Die einzelnen Stile sollen kurz vorgestellt und diskutiert werden.

Geschichteter Architekturstil. Der geschichtete Architekturstil, z. B. die Verwendung von Remote Procedure Calls (RPCs), eignet sich sehr gut für die Integration von Smart Objects in existierende Infrastrukturen: Multi-Tier Systeme, wie in IT-Infrastukturen dominierend, verfolgen meist einen geschichteten Ansatz. Allerdings hat sich dieser Architekturstil als nicht effizient für den direkten Einsatz bei Smart Objects herausgestellt (siehe [4]).

Objektorientierter Architekturstil. Die Zugriffssemantik in datenzentrierten Architekturen sind ähnlich dem geschichteten Architekturstil: Sie sind referenziell eng gekoppelt und erfordern Garantien über die Verfügbarkeit von Ressourcen, um effektiv arbeiten zu können. Diese sind, wie oben beschrieben, bei Smart Objects nur eingeschränkt gegeben. Beispiele für datenzentrierte Architekturstile sind dateiorientierte Systeme und Kopplungen, wie etwa in [14] implementiert. Zum objektorientierten Architekturstil lassen sich auch SOAP basierte Systeme zuordnen. SOAP bzw. das darauf aufbauende UPnP wird in einigen Sensorsystemen, z. B. Infineon's Sindrion, als Architekturstil verwendet. Um eine Implementierung effizient zu ermöglichen, werden in Sindrion allerdings Teile der Funktionalität serverbasiert verarbeitet.

Ereignisbasierter Architekturstil. Dieser Architekturstil wird bei der Kopplung von Smart Objects an IT-Systeme oft eingesetzt. Darunter fallen Publish/Subscribe Systeme, und kontextsensitive, strengtypisierte ereignisbasierte Kommunikationssemantik von RAUM [15]. Dieser Architekturstil erlaubt eine weitgehende referenzielle Entkoppelung zwischen den beteiligten Computersystemen und ist in Sensornetzwerken weit verbreitet.

Shared Data-Space Architekturstil. In diesem Architekturstil werden Ressourcen als geteilte Datenräume aufgefasst. Die referenzielle Entkopplung erfolgt dadurch, dass keine feste Bindung, sondern ein Anfragestil verwendet wird. Dieser Architekturstil eignet sich ebenfalls sehr gut für die Kopplung von IT-Infrastrukturen und Smart Objects. Beispiele sind TinyDB [5], die eine SQL-ähnlich Zugriffssemantik unterstützt, und Blackboard-orientierte Systeme wie Lime [12].

3.1 Active Interfaces

Die zuvor aufgeführten Koordinations- und Architekturstile sind sehr unterschiedlich ausgeprägt. Erschwerend kommt hinzu, dass die konkreten Implementierungen weitgehend inkompatibel sind, sowohl was ihre Kommunikationssemantik als auch was die verwendeten Datenstrukturen anbelangt. In der Praxis müssen aber unter-

schiedliche Technologien und somit auch unterschiedliche Architekturstile unterstützt werden. So wurden beispielsweise im EU-Projekt CoBIs drei Technologieplattformen verwendet: Particle Computer (U. Karlsruhe/Particle Computer), uNodes (U. Twente/Ambient) und Sindrion (Infineon). Während die höheren Schichten des auf der Particle Platform laufenden AwareCon Netzwerkstacks eine Kombination eines Shared Data-Space und Event-Architekturstils darstellen, benutzt Sindrion einen objektorientierten Ansatz (UPnP). Das verwendete IT-Backend verwendet dagegen einen geschichteten Architekturstil. Um Heterogenität zwischen den Smart Objects und der Infrastruktur zu gewährleisten, könnten die innerhalb einer Anwendung eingesetzten Sensorsysteme gezwungen sein, alle möglichen Kommunikations- und Koordinationsarten zu unterstützen. Eine solche Konstruktion würde allerdings eine sehr komplexe Umsetzungslogik zwischen Smart Object-Kommunikation und zur IT-Infrastruktur hin erfordern.

Unser Vorschlag ist statt dessen, Active Interfaces zu verwenden: Ein Active Interface beinhaltet neben der Schnittstelle selbst eine Beschreibung, wie die dort aufgeführten Informationen angesprochen, interpretiert und verarbeitet werden sollen. Active Interfaces können dazu verwendet werden, automatisch sogenannte Transformations-Stubs zu generieren, die zwischen Smart Objects und IT-Services gelagert sind und mittels eines Proxies angepasste Schnittstellen sowohl für die IT-Infrastruktur als auch für die Smart Objects zur Verfügung stellen (Bild 1). Solche Active Interfaces sind damit dynamische Mittler zwischen beiden Systemen: Die Möglichkeit der automatischen Generierung erlaubt eine weitgehende referenzielle und semantische Entkopplung auf Seiten der Smart Objects, bei gleichzeitiger Kopplung an IT-Infrastrukturen. Active Interfaces bestehen aus zwei parallelen, dynamisch durchgeführten Transformationsabläufen (Bild 2): Zum einen werden im Backend verzeichnete Dienstbeschreibungen durch syntaktische Transformation in Gateway-Schnittstellen umgeformt. Zum anderen wird durch semantische Transformation eine Schnittstelle zwischen CoBIs Gateway und den Sensornetzwerken erstellt. Konkret werden solche Active Interfaces als dynamische Proxies implementiert, die auf der IT-Infrastruktur zugewandten Seite einen IP-basierten Dienstzugriff anbieten, auf der anderen Seite das entsprechende Smart Object Koordinationsprotokoll besitzen.

3.2 Logic-Push: Integration und Weiterleitung der Geschäftslogik

Eine Integration von IT-Ablauflogik und der von den Smart-Objekts verwendeten Ablauflogik ist für viele Anwendungen notwendig, um dem Anwender Garantien über den Ablauf geben zu können. In Geschäftsprozessen werden diese Garantien und die dazugehörige Logik oft explizit mit Hilfe und innerhalb des IT-Systems spezifiziert. Das IT-System setzt dann die Anforderungen in entsprechende Abläufe um und überwacht diese.

Werden Smart Objets in solche Geschäftsabläufe integriert, müssen deren Ablauflogiken ebenfalls integriert werden. Wie zuvor beschrieben gibt es dafür zwei Möglichkeiten: Die Verwendung einer fixen Ablauflogik der Smart Objects und Integration der von dort gelieferten Sensordaten in den existierenden Geschäftsprozess, oder die Integration von Smart Objects als Teil der Ablauflogik in den Geschäftsprozess selbst.

Innerhalb des hier vorgestellten Systems wurden Smart Objects zu einem Bestandteil des Geschäftsprozesses, indem die IT-Infrastruktur Teile der Geschäftslogik an die Smart Objects zur autonomen Abarbeitung weiterleitet und damit dauerhaft auslagert. Dies geschieht durch die Weiterleitung von *Event-Condition-Action* (ECA) Regeln vom IT-Backend-System zu den Sensorknoten hin bzw. deren Aufteilung zwischen den Sensorknoten. Die Ereignisse und Aktionen müssen anwendungsspezifisch festgelegt werden. Als Bedingungen erlaubt das System einfache numerische und musterbasierte Vergleiche. In Bild 3 ist eine Aufgabe zu se-

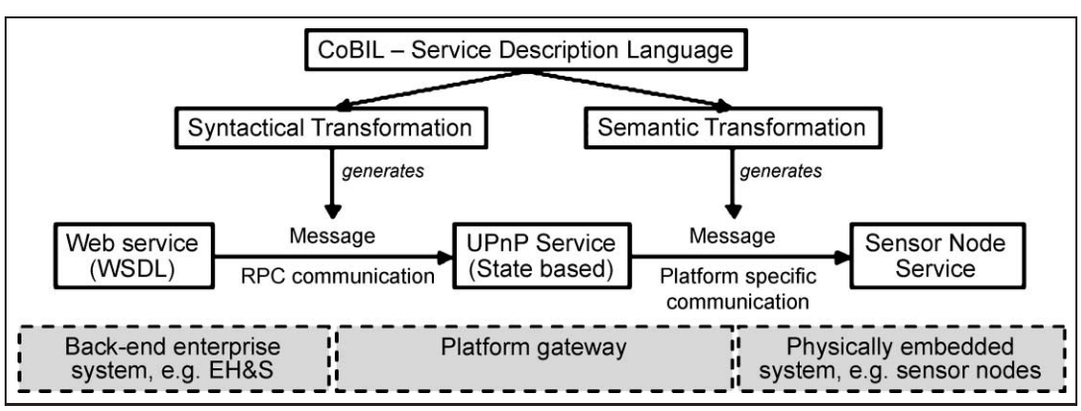

Bild 2 Transformation von Dienstbeschreibung auf Gateway-Dienste und Sensorknoten.

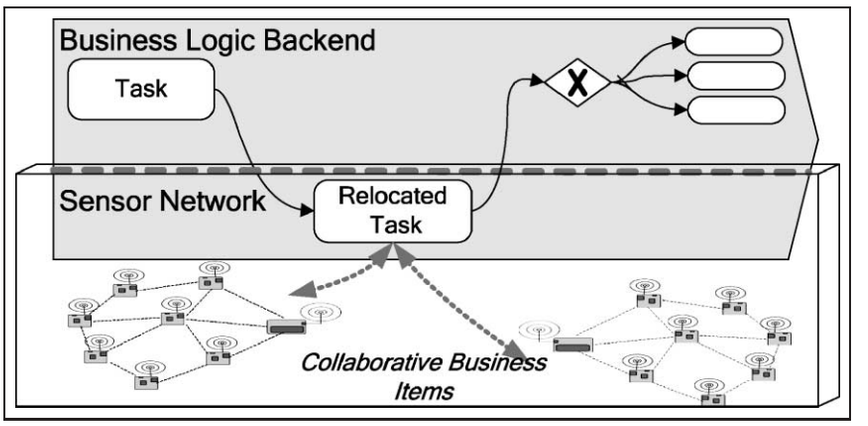

Bild 3 Relokation (Push) der Ablauflogik von der IT-Infrastruktur auf Smart Objects.

hen, die von der IT-Infrastruktur hin zu den Smart Objects ausgelagert wurde. Die Koordination zwischen dem (existierenden) IT-Infrastruktur Prozess und dem Smart Object findet mittels Benachrichtigung (*Events*) bzw. durch Abfrage von Dienstleistung (*Services*) statt. Jenseits dieser Kommunikation, arbeitet das Smart Object autonom bzw. in Kollaboration mit anderen Smart Objects.

3.3 Service Discovery

Um eine effektive Einbindung von Smart Objects in IT-Infrastrukturen zu gewährleisten, ist es notwendig, einen Mechanismus bereitzustellen, der Informationen und Dienste der Smart Objects anbieten kann. Dieser Vorgang wird mit dem englischen Wort Service Discovery bezeichnet. Generell existieren zwei Ansätze für die Realisierung von Service Discovery: Im infrastrukturlosen, nachfrageorientierten Ansatz werden durch Multicast oder vorherige Kenntnis der Adresse Dienste eines Rechnersystems angefragt. Im zweiten, infrastrukturgetriebenen Fall wird ein Verzeichnisdienst verwendet, um Dienste zu finden. Ein Beispiel für verzeichnisorientiertes Service Discovery ist der UDDI Registrierungsdienst, wie er im Web Service Interoperability Standard beschrieben ist. Solche verzeichnisorientierten Dienste fügen sich sehr gut in bestehende IT-Infrastrukturen ein, sind allerdings von der Dynamik der Veränderungen der Zuordnungen, wie sie für Smart Objects typisch sind, überfordert.

Infrastrukturlose, Plug-and-Play orientierte Ansätze, wie sie z. B. in UPnP standardisiert wurden, sind hier durch ihre weitgehende referenzielle und semantische Entkopplung besser geeignet. Solche Multicast-orientierten Systeme eignen sich sehr gut für die Echtzeit-Verfügbarkeitsabfrage von Diensten in Smart Object orientierten Systemen.

4 CoBIs Gateway Architektur

In diesem Abschnitt wird eine konkrete Implementierung einer Anbindung von Smart Objects an eine IT-Infrastruktur beschrieben. Dieses System wurde im Zug des EU-Projekts CoBIs (*Collaborative Business Items*) entworfen, implementiert und getestet. Die Architektur des Systems besteht aus drei Ebenen: Den Business Services der IT-Infrastruktur, den Smart Object Frontends, sowie einer dazwischen gelagerten, dynamisch aufgebauten Service Proxy-Architekturschicht.

Die Gateway-Schicht implementiert eine UPnP-Schnittstelle zur IT-Infrastruktur hin und erlaubt so einen flexiblen und standardisierten Zugang zu Smart Objects. Zwischen den Smart Objects werden weiterhin die nativen, für den Einsatzzweck optimierten Kommunikationsarchitekturen verwendet. Die dafür notwendige Transformation wird im Gateway durchgeführt. Bild 4 zeigt ein Beispiel einer Transformation zwischen Gateway und dem Particle Sensornetzwerk. Die Funktionsmechanik des semantischen Teils der Transformation ist spezifisch für das angekoppelte Sensornetzwerk und muss beim Entwurf des Sensornetzwerks erstellt werden. Die konkrete Ankopplung des Sensornetzwerks kann dann dynamisch basierend auf den spezifisch für jedes Sensornetzwerk erstellten Interface-Primitiven konfiguriert werden.

Der UPnP Standard wurde als Schnittstelle zwischen Gateway und IT-Infrastruktur gewählt, da er eine leichtgewichtige, infrastrukturlose, aber dennoch umfassende Implementierung von dienstorientierten Systemen ermöglicht. UPnP umfasst das *Simple Object Access Protocol* (SOAP) zum einfachen Zugriff auf Dienste und Objekte, das *Simple Service Discovery Protocol* (SSDP) für das Service Discovery und die *General Event Notification Architecture* (GENA), die ein Koordinationsprotokoll für Eventsemantik umfasst. Das auf IP aufbauende UPnP ist in zahlreichen Betriebssystemen wie z. B. Microsoft Windows und zahlreichen Linux-Derivaten bereits eingebettet, und kann deshalb transparent von IT-Infrastrukturen verwendet werden.

Ein Schlüsselelement der beschriebenen Architektur ist die Implementierung von Active Interfaces durch dynamisch generierte Proxies. Diese Proxies besitzen native UPnP Schnittstellen, und propagieren darüber detaillierte Dienstbeschreibungen für die implementierte Funktionalität. Die Proxies selbst existieren wiederum nur als virtuelle Repräsentation der Dienste: Anfragen an die Proxies werden basierend auf der Logik der Active Interfaces vom Gateway transformiert und an das Smart Objects Sensornetzwerk weitergegeben. Dies ermöglicht die gleichzeitige Einbindung verschiedener heterogener Smart Object-Typen innerhalb einer IT-Infrastruktur, sowie deren nahtlose Integration in existierende IT Geschäftsanwendungen.

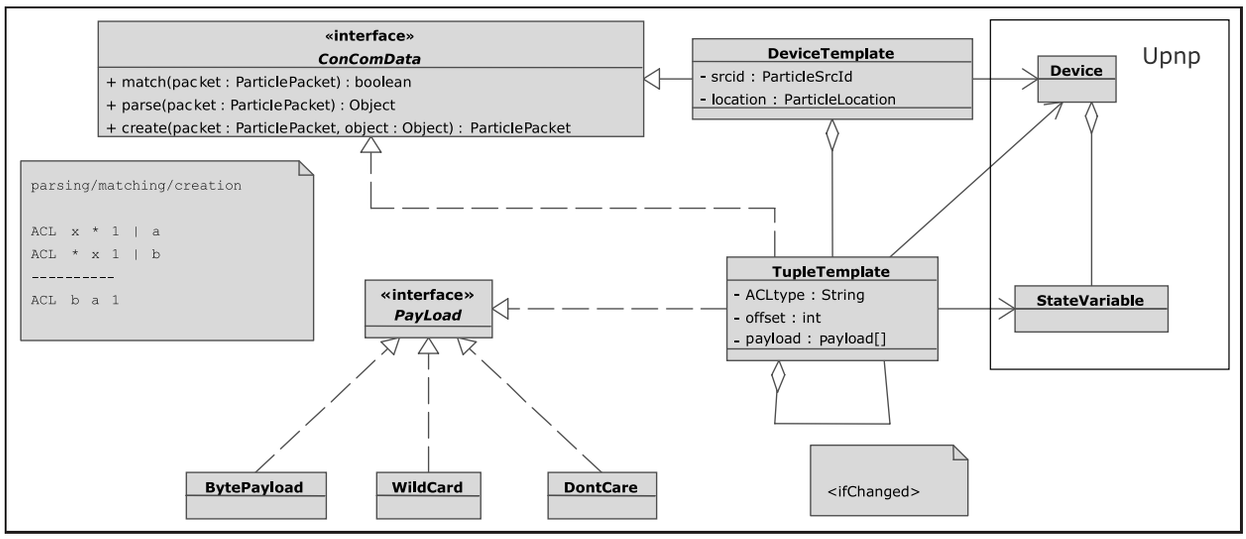

Bild 4 Beispiel einer Transformation einer RPC Aufrufsemantik auf das ereignisbasierte Particle Sensornetzwerkprotokoll.

Gateway: Initialisierung und Primitive

Das Gateway der Plattform kapselt alles Domänenwissen, das für die Kommunikation zwischen IT-Infrastruktur und Smart Objects benötigt wird. Es lassen sich vier grundlegende Typen von Dienstinstallationen unterscheiden: Im ersten Fall leitet das Gateway vordefinierte „hello“ und „bye-bye“ Pakete weiter (Bild 5A). Alternativ kann aktive Nachfrage alle Dienste dazu veranlassen, „hello“ Pakete als Antwort auf „Ping“ Pakete zu senden (Bild 5B). Eine passive Variante (Bild 5C) verwendet statt spezieller „hello“ Pakete Dienstidentifikatoren in allen Paketen. Eine letzte, allerdings ungünstige weil statische Methode ist die permanente Vorinstallation von Diensten.

Das hier vorgestellte System unterstützt sieben verschiedene Sendeprimitive zur Kommunikation zwischen Smart Objects und der IT-Infrastruktur:

- Nicht-blockierendes Senden, Empfangen und Dienstaufruf,
- Blockierendes Senden, Empfangen und Dienstaufruf,
- Rückrufaufforderung.

Diese Primitive werden auf entsprechende UPnP Aufrufe (SOAP) abgebildet.

5 Feldversuch

Die Implementierung des gesamten Systems wurde innerhalb von BP's größter Acetyl Produktionsstätte vorgenommen. Die hier vorgestellten Ergebnisse stammen aus einem Versuchslauf zwischen dem 13. November und dem 15. Dezember 2006. Der Feldversuch umfasste die Ausstattung von 21 Chemie-Containern mit eingebetteter Miniaturelektronik, die somit zu Smart Objects erweitert wurden. Als Realisierungsplattform wurden Particle Computer Sensorknoten verwendet. Das Ziel ist, Lagerbedingungen für die Chemikalien zu überwachen. Die Chemie-Container (siehe Bild 6) werden vielfach innerhalb der Lagerstätte bewegt.

Folgende Regeln wurden innerhalb des Systems implementiert:

- Grenzwert für maximale Anzahl von Containern innerhalb einer Fläche,
- Inkompatibilität von Lagerbedingungen basierend auf Zusammenhängen zwischen gelagerten Chemikalien, Lagerort und Lagerabstand,
- Einhalten von Umgebungsbedingungen, hier vereinfacht nur maximale und minimale Temperatur,
- Maximale Lagerzeit pro Objekt.

Diese Logik wurde innerhalb des Geschäftlogik-Moduls SAP EH&S (*Environment Health and Safety*) innerhalb der existierenden IT-Infrastruktur implementiert. Diese von den Smart Objects zu beachtende Logik der Objektüberwachung wurde dann vom IT-

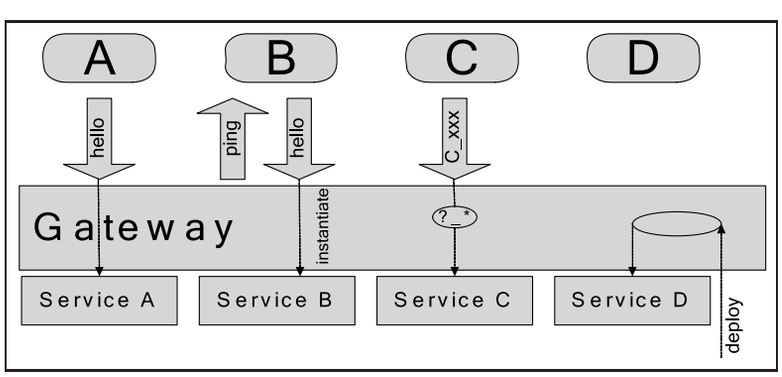

Bild 5 Koordination IT-System dem Gateway.

Bild 6 BP Chemielager in Hull, UK.

System als ECA-Regeln auf die Smart Objects heruntergeladen. Ein Regelabgleich erfolgte, sobald neue Regeln zur Verfügung standen und eine stabile Netzwerkverbindung zugesichert werden konnte.

Zusätzlich wurde ein Lokationssystem, sowie eine technische Gateway-Infrastruktur (Linux MIPS basiertes System mit 200 MHz, sowie ein Linux Intel Pentium System mit 800 MHz) installiert. Über ersteres System war es den Smart Objects möglich, ihre Position auf etwa einen Quadratmeter genau zu bestimmen. Über zweiteres konnten die Smart Objects im Falle Verbindung mit der IT-Infrastruktur aufnehmen. Die Gateway-Infrastruktur bildete auch den technischen Host für die oben beschriebenen Proxy-Objekte der Active Interfaces.

Versuchsdurchführung

Der Versuch offenbarte erhebliche Störungen beim Betrieb einer technischen Infrastruktur innerhalb der BP Produktionsstätte. Davon war insbesondere die kabellose Kommunikation zwischen den Computersystemen betroffen, unter anderem die 802.11 (Wavelan) Verbindung der IT-Infrastruktur.

Diese Störungen hatten erhebliche Auswirkungen auf den Bereich der traditionellen IT-Dienste, insbesondere der auf ungesicherten Protokollen basierenden Dienste UDP and DHCP. Hingegen war die Arbeit der Smart Objects durch die Verwendung der autonomen Durchführung von Anwendungslogik und deren lokale Kommunikation nicht betroffen, sodass Warnungsmeldungen und ähnliches spezifikationsgemäß in Echtzeit ausgegeben werden konnten. Bild 7 zeigt die durchschnittliche Last pro Meldung. Während die Grundlast von den regelmäßigen Statusreports zur IT-Infrastruktur bestimmt ist, ist auch eine Überlastspitze bei der Kommunikation zwischen Smart Objects und IT-Infrastruktur zu erkennen (Markierung). Diese tritt in Alarmsituationen durch die vermehrte Anzahl von Ereignissen ein. Diese hat aber keine Auswirkung auf die Funktionalität, da durch die lokale Abarbeitung der IT Prozesslogik auf dem Smart Object eine Latenzzeit im Millisekundenbereich zugesichert werden kann. Bild 8 zeigt, dass durch diese lokale Verarbeitung der Information das Backend-System erheblich entlastet werden konnte. Die Anzahl der lokalen Meldungen der Smart Objects ergibt sich aus den Anforderungen der Anwendungen, die in den ECA Regeln festgelegt sind und ist applikationsspezifisch. Durch die hohe Lokalität der Kommunikation konzentriert sich die Kommunikation zum Backend aber auf die Auslieferung von Ereignismeldungen und Zusammenfassungen der Sensordaten.

Um einen Vergleich zu einem alternativen, zentralen Systemansatz zu testen, wurde die Anzahl der Kommunikationspakete künstlich erhöht, um die Auslieferung aller Sensorinformation an das IT-Backend-System zu simulieren. Am Test beteiligt waren ein SAP EH&S Netweaver System (SAP Web Application Server, WebDynPro frontend mit WebServices, 1.6 GHz, 2 GB Hauptspeicher) sowie 20 Smart Objects. Im Zeitraum des Tests wurden durchschnittlich etwa 200 Meldungen pro Minute an das Backend-System ausgeliefert. Dadurch entstand eine Überlastsituation, die zu erheblichen Latenzzeiten und nach 90 Minuten zum Absturz des Systems führte. Obwohl das verwendete Backend-System sicher am unteren Leistungsbereich anzusiedeln ist, kann das Ergebnis als übertragbar angesehen werden, da auch die Anzahl beteiligter Smart Objects sehr gering ist.

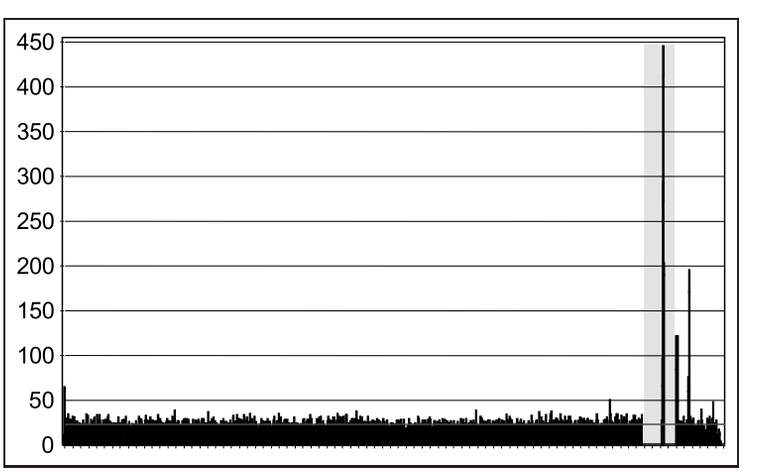

Bild 7 Anzahl Pakete Smart Objects zur IT-Infrastruktur.

Parameter	Wert
Durchschnittliche Last auf Anwendungsserver	7 Meldungen/Minute
Spitzenlast auf Anwendungsserver	212 Meldungen/Minute
Durchschnittliche Last im Netzwerk vor Gateway	187 Meldungen/Minute
Verhältnis lokale Netzwerklast / Last auf Anwendungsserver	96% lokal verarbeitet
Durchschnittliche Latenz Propagierung neuer Regeln und Konfigurationen	23.3s pro Smart Object

Bild 8 Anwendungsserver-, Netzwerklast und Latenzzeit im Feldversuch.

6 Zusammenfassung

Die Zusammenarbeit zwischen Smart Objects und existierender IT-Infrastruktur erfordert es, ein über die reine Anbindung der Smart Objects als Sensor-Informationsquellen hinausgehendes Konzept zu entwickeln. Dieser Bericht stellt mit Active Interfaces und dem Push von Ablauflogik auf die Smart Objects zwei Methoden vor, die die Integration von Smart Objects in IT-Infrastrukturen erleichtert bzw. erst ermöglicht. Wir konnten innerhalb einer Implementierung in einem Testlauf in einer Produktivumgebung bei BP in Großbritannien nachweisen, dass diese Mechanismen nicht nur die Integration zwischen beiden Informationssystemwelten unterstützen, sondern auch den Ablauf von Anwendungen erheblich robuster gestalten kann. Im Laufe des Testbetriebs konnte auch gezeigt werden, dass ein alternativer zentralisierter Systemansatz zur Einbindung von Sensornetzwerken ungeeignet ist. Die erzielten Ergebnisse empfehlen den vorgestellten Ansatz als technologische Blaupause für den Einsatz von massengedruckter Einfachelektronik in zukünftigen IT-Infrastrukturen.

Literatur

[1] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray. Cooperative Artefacts: Assessing Real World Situations with Embedded Technology. Ubicomp 2004.

[2] C. Bornhövd, T. Lin, S. Haller, and J. Schaper. Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-ID Infrastructure. In: Proc. of 30th VLDB Conference, Toronto, Canada.

[3] Z. Nochta, N. Oertel, and P. Spiess. Relocatable Services and Service Classification Scheme. CoBIs Deliverable Report, http://www.cobis-online.de/files/Deliverable_D101.pdf, 2005.

[4] U. Saif and D. J. Greaves. Communication Primitives for Ubiquitous Computing or RPC Considered Harmful. In: 21st ICDCSW, p. 0240, 2001.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. In: ACM TODS, 2005.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In: ASPLOS-IX, 2000.

[7] P. Havinga. The Quest for Low Cost, Ultra Low Power Wireless Networks for smart environments, Ambient Systems. white paper, 2006.

[8] C. Decker, A. Krohn, M. Beigl, and T. Zimmer. The Particle Computer System. In: Proc. of the ACM/IEEE Fourth Int'l Conf. on Information Processing in Sensor Networks, Los Angeles, 2005.

[9] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Harvard Technical Report TR-21-04, 2004.

[10] S. Madden, M.Franklin, J. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. In: ACM TODS, 2005.

[11] Universal Plug and Play Device Architecture, Microsoft Corportation, 1999.

[12] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A Coordination Middleware Supporting Mobility of Hosts and Agents. In: ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 15, no. 3, pp. 279–328, 2006.

[13] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. In: Wireless Networks, March 2002.

[14] C. Decker, T. Riedel, M. Beigl, and A. Krohn. A file system for system programming in ubiquitous computing. In: Personal and Ubiquitous Computing Journal, Vol. 11, Issue 1, pp. 21–31, Springer Verlag, 2007.

[15] F. Hupfeld and M. Beigl. Spatially aware local communication in the RAUM system. In Proc. of the IDMS, Enschede, The Netherlands, Oct 2000, pp. 285–296.


1


2


3

1 Prof. Michael Beigl ist Professor für Verteilte und Ubiquitäre Systeme an der Carl-Friedrich-Gauß-Fakultät der Technischen Universität Braunschweig. Sein Forschungsbereich umfasst Systemdesign, Netzwerke, Protokolle und Sicherheitsaspekte für mobile, vernetzte, smarte Sensorsysteme. Im Bereich Ubiquitous Computing beschäftigt sich Michael Beigl mit innovativen neuen Systemansätzen und Appliances sowie deren Anwendung. Ein weiterer Bereich seiner Forschung ist die Integration von miniaturisierten Computersystemen in existierende verteilte Systeme, insbesondere ERP-, SOA-, Web-basierte und Virtual Reality-Systeme.
Adresse: Technische Universität Braunschweig, Institut für Betriebssysteme und Rechnerverbund, Mühlenpfordtstraße 23, 38106 Braunschweig, Deutschland, E-Mail: beigl@ibr.cs.tu-bs.de

2 Christian Decker ist wissenschaftlicher Mitarbeiter am Telecooperation Office (TecO) der Universität Karlsruhe. Er ist an verschiedenen Projekten zur Anwendung von Ubiquitous und Pervasive Computing in Industrie- und Geschäftsanwendungen beteiligt. Gemeinsam mit Till Riedel war er für die Implementierung von Smart Objects im CoBIs Projekt verantwortlich.
Adresse: TecO, Universität Karlsruhe, Vincenz-Prießnitz-Str.1, 76131 Karlsruhe, Deutschland, E-Mail: cdecker@teco.edu

3 Till Riedel ist wissenschaftlicher Mitarbeiter am Telecooperation Office (TecO) der Universität Karlsruhe. Seine Forschungen umfassen neuartige Softwaremechanismen für eingebettete Sensorsysteme. Innerhalb von CoBIs war Till Riedel verantwortlich für Entwurf und Implementierung von kollaborativen SOA-Konzepten für Smart Objects.
Adresse: TecO, Universität Karlsruhe, Vincenz-Prießnitz-Str.1, 76131 Karlsruhe, Deutschland, E-Mail: riedel@teco.edu