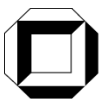
Universität Karlsruhe - Fakultät für Informatik - Bibliothek - Postfach 6980 - 76128 Karlsruhe

Mobile und Verteilte Systeme
Ubiquitous Computing
Teil V
Seminar im WS 2007/08

Till Riedel, Martin Berchtold, Patrik Spieß,
Luciana Moreira Sa de Souza, Christian Decker

Interner Bericht 2008-2



Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

ISSN 1432-7864

Fakultät für **Informatik**

Vorwort

Das Seminar „Mobile und verteilte Systeme – Ubiquitous Computing“ am Telecooperation Office (TecO) erfreut sich großer Beliebtheit. Der vorliegende interne Bericht enthält Beiträge von Studierenden zu diesem Seminar, das im WS 2007/08 in dieser Form zum fünften Mal stattgefunden hat.

Die Themenauswahl für das Seminar orientiert sich im wesentlichen an aktuellen wissenschaftlichen Fragestellungen in den Bereichen:

- Technologien für ein Internet der Dinge
- Sensorverarbeitung und Kontexterkennung
- Softwaretechnik für ubiquitären Informationssystemen

Aufgrund des stetig wachsenden Interesses der Studierenden an diesen Themenbereichen haben wir uns entschlossen einen Seminarband mit den Beiträgen unserer Studierenden als internen Bericht zu veröffentlichen. Durch die engagierte Mitarbeit der beteiligten Studierenden wird ein Ausschnitt aus diesem komplexen und umfassenden Themengebiet klar und übersichtlich präsentiert. Für den Fleiß und das Engagement unserer Seminarteilnehmer wollen wir uns an dieser Stelle daher herzlich bedanken.

Bestärkt durch die gute Resonanz der Studierenden, werden wir dieses Seminar auch im nächsten Wintersemester – dann natürlich mit aktualisierten Themen – wieder anbieten.

Karlsruhe, Februar 2008

Till Riedel, Martin Berchtold, Patrik Spieß, Luciana Moreira Sá de Souza,
Christian Decker

Inhalt

Komponenten des Polymercomputers

Micha Borrmann

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer: Martin Berchtold
08.01.2008

1 Einleitung

Während man heute mit dem Begriff Computer noch hauptsächlich Personalcomputer assoziiert, so wird erwartet, dass sich dies in der Zukunft ändert. Neue Anwendungen wie "intelligente" Kleidungsstücke (sog. wearable Computers), elektronischer Papierersatz (ePaper), "intelligente" Kühlschränke oder Einkaufswagen sind Stichworte, die den möglichen Einsatz von Computern in der Zukunft beschreiben. Dieser lässt sich im Wesentlich durch zwei Punkte charakterisieren:

1. Kleinstcomputer als Massenelektronik: Integration von Kleinstcomputern in den Alltag bedeutet, dass diese in sehr hohen Stückzahlen zu Verfügung stehen müssen. Für einen Kühlschrank, der sich seines Inhalts bewusst ist oder den "intelligenten" Einkaufswagen müssten beispielsweise sämtliche auf heutigen Verpackungen vorhandene Barcodes durch Funketiketten ersetzt werden. Um eine solche massenhafte Anwendung realisieren zu können, müssen die Kosten jedoch noch drastisch gesenkt werden: Passive RFID Etiketten kosten heute ca. 30 Cent pro Stück - ein solches Etikett auf einer Milchtüte wäre nahezu genauso teuer wie die eigentliche Ware selbst [28].
2. Computer sollen nicht als solche wahrgenommen werden, sondern in die Lebensumgebung integriert sein. Dies erfordert neue Materialien: Für in Kleidung integrierte Kleinstrechner, ePaper oder Leuchttapeten spielt Flexibilität eine entscheidende Rolle. Soll Kleidung Sensordaten über ihren Träger sammeln, so erfordert das neuartige Sensoren. Die Integration von Computern in die Lebensumgebung verlangt mitunter auch neue Schnittstellen in der Mensch-Maschine Kommunikation.

Man setzt große Hoffnungen darauf, dass diese Anforderungen in Zukunft mithilfe von leitfähigen Polymeren zumindest teilweise erfüllt werden können.

2 Leitende Polymere

2.1 Einführung

In dieser Arbeit soll nur kurz auf die chemischen Hintergründe leitender Polymere eingegangen werden. Für eine detailliertere Diskussion sei hier auf [1] verwiesen.

Als das erste Polymer mit nennenswerter elektrischer Leitungsfähigkeiten fand man Polyacethylen (Abb. 1 links). Dieses ansonsten isolierende Polymer zeigte nach Oxidation (Dotierung, s.u.) eine Elektrische Leitfähigkeit [1, 2].

Charakteristisch für leitende Polymere ist ein quasi eindimensionales Backbone System, an den entlang sich Ladungsträger bewegen können. Dieser Backbone besteht meistens aus alternierenden Einfach- und Doppelbindungen, sog. konjungierten Doppelbindungen. In einer solchen alternierdenden Kette treten die Elektronen nicht mehr fest gebunden, sondern delokalisiert (Ladungswolken) auf. Im Orbitalmodell spricht man hierbei von einer Ausbildung von π-Orbitalen [1]. Viele weitere leitende Polymere besitzen die Struktur alternierdender Einfach- und Doppelbindungen analog zu Polyacethylen, oft sind diese jedoch als Ringe alternierender Bindungen ausgebildet (Abb. 1 rechts) [2].

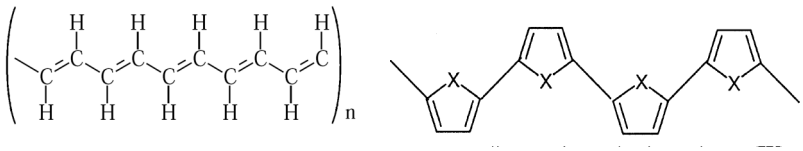


Abbildung 1. Links: Ausschnitt aus einer Polyacethylenkette. Charakteristisch die alternierenden Doppel- und Einfachbindungen [1], rechts: Heterozyklische Organische Polymere: Polyfuran (X = O), Polythiopen (X = S), Polyfuran (X= N-H) [2].

Analog zur Dotierung (engl. "Doping") in anorganischen Materialien existiert dieser Begriff auch für Polymere. Hier bezeichnet er die Reduktion oder Oxidation (chemisch oder über elektromagnetische Felder) von an und für sich isolierenden Polymeren. Werden bei einer Oxidation Elektronen entfernt, so entstehen Radikale, welche wiederum mit Elektronen aus Doppelbindungen rekombinieren können. Dadurch enstehen positiv geladene Störstellen entlang des Backbones, welche unter Einfluss eines elektrischen Feldes entlang der Moleküle wandern und zum Stromfluss beitragen. Für eine detaillierte Diskussion sei auch hier auf [1] verwiesen.

2.2 Eigenschaften leitender Polymere

Elektrische Eigenschaften

Die erreichbaren Schaltgeschwindigkeiten elektronischer Schaltungen hängen proportional von der Ladungsträgerbeweglichkeit im Leiter ab. Diese ist bei leitenden Polymeren jedoch noch wesentlich geringer als bei anorganischen Materialien. Mit Polymeren erreicht man Ladungsträgerbeweglichkeiten im Rahmen von $0,01\ \mathrm{cm}^2/\mathrm{Vs} - 0,02\ \mathrm{cm}^2/\mathrm{Vs}$ [5], erst allmählich dringt man in den Bereich von $1\mathrm{cm}^2/\mathrm{Vs}$ (vergleichbar mit amorphem Silizium) vor [3], die Ladungsträgerbeweglichkeit in einkristallinem Silizium erreicht hingegen Größenordnungen um

1000 cm^2/Vs [4].
Dies hat weitreichende Auswirkungen: Für Hochfrequente Anwendungen (bspw. RFID mit den zugehörigen Frequenzen von 125 Khz oder 13,56MHz) sind heutige Polymere nicht geeignet. Hierfür wären Ladungsträgerbeweglichkeiten in Größenordnungen von amorphem Silizium und mehr nötig [3].

Eine weitere Schwierigkeit besteht im Aufbau komplementärer Schaltungen. Während in anorganischen Schaltungen die CMOS-Technologie (Komplementäre Schaltungen aus p- / und n- Halbleitern) schnelle Schaltfrequenzen erlaubt und dafür sorgt, dass nur in Schaltmomenten ein wesentlicher Anteil an Strom fließt, so ist eine solche komplementäre Technologie mit polymeren Materialen bisher nicht realisierbar. Während p-Halbleiter aus Polymeren schon heute existieren, so ist es derzeit noch schwierig polymere n-Halbleiter mit vergleichbaren Eigenschaften zu produzieren (Probleme beim Ladungsträgertransport und der chemischen Stabilität) [3].

Physikalische Eigenschaften

Die leitenden Polymere reagieren sehr empfindlich auf äussere Einflüsse. So besitzen die Polymere eine relativ offene Struktur (Abb. 2), in welche Gase und Luftfeuchtigkeit aus der Umgebung leicht eindringen können. Die Interaktion der Polymere mit diesen Stoffen sorgt für Änderungen der (elektrischen) Eigenschaften des Materials und verkürzen somit die Lebensdauer organischer Schaltungen. Solche Effekte sind in anorganischen Schaltungen wesentlich geringer, da die Oberflächenschichten anorganischer Materialien sehr wenig durchlässig sind. Interagiert ein leitendes Polymer mit einem Gas, so kann dies als eine sekundäre Dotierung des Materials aufgefasst werden und kann neben elektrischen Eigenschaften auch magnetische und optische Eigenschaften des Polymers beeinflussen. Desweiteren kann es zu einem Ladungsaustausch zwischen Gasen und Polymer kommen, welcher die elektrischen Eigenschaften des Polymers weiter verändert [6].
Bisher bekannte Barrierematerialen reichen nicht aus, daher müssen für organische Schaltungen neuartige Barrieren gefunden werden, die wesentlich undurchlässiger sind. Es existieren Verfahren für die gezielte Entwicklung solcher Barrierschichten, jedoch sind diese nicht für solche hohen Undurchlässigkeiten geeignet [7].

3 Produktionsmethoden von Polymerschaltungen

3.1 Motivation

Dass Polymere trotz ihrer elektrisch schlechteren Eigenschaften im Vergleich zu Silizium im Interesse der Industrie stehen, liegt an den neuen Produktionsverfahren die hiermit möglich werden. Gegenüber den bisher etablierten Technologien besitzen Polymere den entscheidenden Vorteil in Lösung verarbeitet

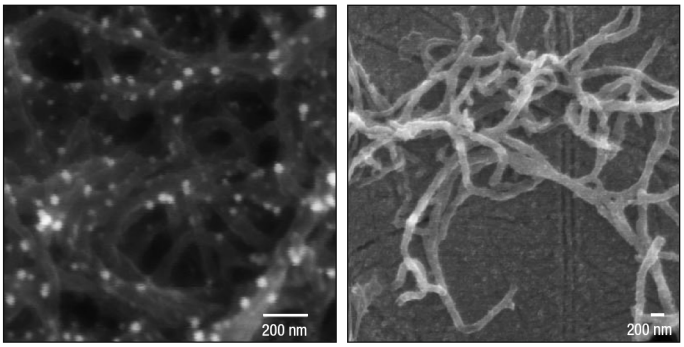


Abbildung 2. Innere Struktur von Polyanilin (links), Oberflächenstruktur von Polyanilin (rechts) [6]

werden zu können [5]. Dies ermöglicht es Polymertinten herzustellen und damit Schaltungen zu drucken. Die Verwendung von Drucktechniken zur Schaltungsherstellung erleichtert eine kontinuierliche Produktion und erlaubt durch höhere Stückzahlen signifikant niedrigere Fertigungskosten als konventionelle Technik. Neben dem Tintenstrahldruck setzt man hierbei auch auf andere Druckverfahren wie Flexo- oder Gravurprinting. Bisher erfolgt die Mikroelektronikherstellung unter hohem Energieaufwand unter Reinraumbedingungen und komplizierter Verfahren. Dabei wird in der Waferherstellung in mehreren Arbeitsschritten das Substrat (Trägermaterial auf welches die Schaltung aufgebracht wird) wiederholt dotiert, oxidiert und einzelne Strukturen weggeäzt (Lithografie). Eine solche Produktion erlaubt nur eine vergleichbar geringe Stückzahl und verursacht dementsprechend hohe Kosten. Dies wird im Kostenvergleich mit Gravurdruck deutlich. Während es heute möglich ist Siliziumwafer mit einem Durchmesser von 30 cm mit einer Stückzahl von 6000 Wafern pro Woche bei einer produzierten Fläche von ungefähr 88000m^2/Jahr zu verarbeiten, ist es mit einer großen Gravurdruckmaschine hingegen möglich auf ein Substrat von 3,80 m Breite bei einer Geschwindigkeit von 15 m/Sekunde zu drucken, und somit eine Fläche von $205,00\text{m}^2$ pro Stunde zu verarbeiten (und somit innerhalb von 26 Minuten die selbe Fläche wie eine Mikroelektronikfabrik in einem Jahr). Vergleicht man dazu noch die Kosten für den Aufbau einer Mikrochipfabrik mit denen einer Gravurdruckmaschine (etwa 4 Milliarden Euro im Vergleich zu 40 Millionen Euro) wird deutlich, warum das Interesse der Industrie an solchen Verfahren so groß ist[1]. Durch die Verwendung von Polymeren wird es weiterhin möglich von den starken Beschränkungen bzgl. der Substrate in der anorganischen Produktion wegzukommen. So sind nun beispielsweise auch flexible Substratmaterialen möglich [3], [7].

Es existieren bereits diverse Massendruckverfahren für den visuellen Druck, wie Gravurprintung, Flexoprinting, Ink Jet Printing, Offsetprinting und weitere. Diese unterscheiden sich in Druckgeschwindigkeit, möglicher Schichtdicke,

verwendbarer Tinten und weiteren Kenndaten. Weiterhin unterscheidet man zwischen dem homogenen Aufbringen von Schichten (Coating) und dem Druck von Strukturen (Patterning). Die Wahl des Druckverfahrens richtet sich dann nach Anforderungen wie Anzahl der Schichten/ Zeit, Schichtdicke und Homogenität der Schichten. In der Massenfertigung erlaubt vor allem Flexoprinting eine sehr hohe Produktionsgeschwindigkeit, Ink Jet Printing erlaubt hingegen das Aufbringen mehrerer Schichten zeitgleich [1].

Auch wenn bereits viele Erfahrungen auf dem Gebiet des Massendrucks vorliegen, so sind diese nicht ohne weitere Anpassungen auf den Druck elektronischer Schaltungen übertragbar, da der traditionelle Druck ein gutes visuelles Ergebnis zum Ziel hat, der Fokus bei Elektronik aber auf den elektrischen Eigenschaften liegt. Um die geforderten Schaltfrequenzen realisieren zu können benötigen diese Verfahren eine weit höherere Auflösung als für den visuellen Druck, da die Schaltgeschindigkeiten von den Strukturgrößen abhängen (s.u.). Das visuelle Ergebnis ist bei der Produktion von elektronischen Schaltkreisen hingegen unwichtig. Auch die Wahl der Lösungsmittel gestaltet sich schwieriger. Lösungsmittel nachfolgender Druckschichten dürfen bereits gedruckte Schichten nicht anlösen. Eine Gegenüberstellung der Anforderungen an die Technologie liefert Abb. 3 [3].

requirement	traditional	electronics
resolution	> 20 µm	<< 20 µm
register	± 5 µm	< 5 µm
edge sharpness	high	very high
layer thickness	~ 1 µm	30 ... 300 nm
homogeneity	not important	very important
adhesion of layers	important	important
solvents of inks	cost issue	functional issue
purity of solutions	not important	very important
visual properties	very important	not important
electronic properties	not important	very important

Abbildung 3. Übersicht über die Anforderungen an den Druck elektronischer Schaltungen gegenüber dem traditionellen Druck [3].

3.2 Möglichkeiten und Grenzen der Drucktechnik

Organische Feldeffekttransistoren als Basis weiterer Schaltungen

Transistoren bilden die Grundlage für weitere Logikgatter in elektrischen Schaltkreisen. Ein Feldeffekttransistor besteht aus einem halbleitenden Kanal zwischen

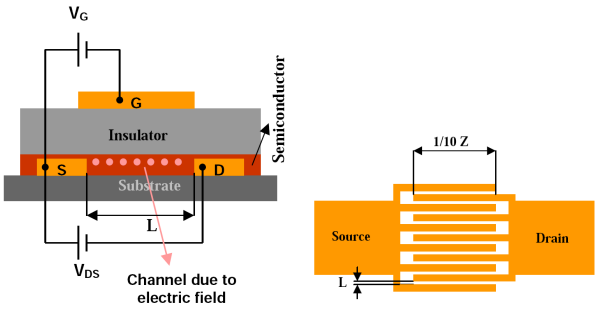


Abbildung 4. Links: Aufbau eines Top-Gate Feldeffekttransistors. Rechts: Typischer Aufbau der Drain-/ Source-Elektroden bei einem Top-Gate OFET. Kanallänge L (Abstand der Elektroden zueinander), Kanalbreite Z (Überlappungsbereich der Elektroden) [1].

zwei Elektroden (Source und Drain) und einer durch ein Dielektrikum vom Kanal isolierten Elektrode (Gate). Über das Potential der Gateelektrode wird ein elektrisches Feld verändert, welches den Stromfluss im halbleitenden Kanal beeinflusst (Abb. 4) [1].

Die erreichbaren Strukturgrößen in der Transistorfertigung haben direkten Einfluss auf die erreichbare Schaltfrequenz: So ist die erreichbare Schaltgeschwindigkeit umgekehrt proportional zum Quadrat der Kanallänge. Für RFID Transponder mit 13,56 Mhz werden hier beispielsweise Größenordnungen um $10\mu m$ benötigt. [3].

Tintenstrahldruck. OFET wurden bereits erfolgreich per Tintenstrahldruck gefertigt [5]. Dabei werden die Nachteile des Tintenstrahldrucks deutlich: Der Flug der Tintentropfen und deren Verteilung auf der Substratoberfläche ist ein statistischer Prozess. Dies macht es unmöglich Source-Drain Strukturen um 10 μm ohne versehentliche Kurzschlüsse zu drucken. Dessen Auflösung ist auf $20-50$ μm beschränkt [5]. Einen möglicher Ausweg hieraus bietet die Vorbehandlung des Substrats. So wurde gezeigt, dass durch das Aufbringen hydrophober Schichten das ungewollte Ausbreiten der Tinte auf dem Substrat verhindert werden kann. Damit konnten Kanallängen von 5 μm erzeugt werden (Abb. 5) [5]. Allerdings hat eine solche Substratvorbehandlung den Nachteil, dass es der Kontinuität in der Produktion entgegensteht und somit die produzierbaren Stückzahlen senkt. Während in [5] für diese Vorbehandlung noch Photolitographie verwendet wurde bieten sich für die Zukunft Verfahren wie Soft-Lithographie oder Photopatterning an, die diesen Nachteil mindern sollen [5].

Flexoprinting. Auch mit Flexoprinting wurden bereits erfolgreich OFET gedruckt. Abbildung 6 zeigt den Kanal eines solchen Transistors mit einer Kanallänge von ca 35 μm.

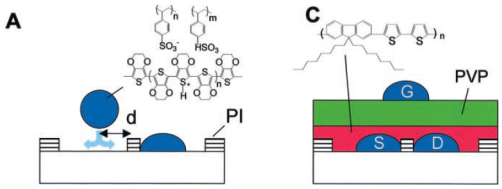


Abbildung 5. Substrat-Pattering erlaubt es, auch mit Tintenstrahldruck Kanallängen von 5 μm zu erzeugen. Hierbei verhindern hydrophobe Barrieren das ungewollte Ausbreiten der Tinte auf dem Substrat. [5]

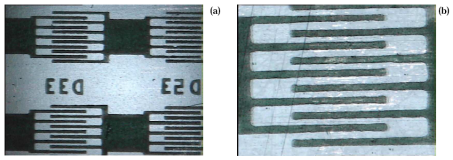


Abbildung 6. Mit Flexoprinting gedruckter OFET (rechts vergrößerter Ausschnitt). Die Kanallänge beträgt ca. 35 μm [1].

Für die erfolgreiche Herstellung von OFET sind neben dem halbleitenden Kanal auch noch Dielektrika nötig, die den Kanal von der Gate-Elektrode trennen. Während bei Silizium hierfür das Silizium oxidiert wird, müssen in der Plastikelektronik hier erst entsprechende Materialen gefunden werden. Für weiterführende Schaltungen (Inverter als Basisgatter) werden ausserdem noch Möglichkeiten zur Durchkontaktierung (Via-Holes) benötigt. Für die organischen Elektronik können die Verfahren der anorganischen Elektronik nicht unverändert übernommen werden. Neue Verfahren, welche der kontinuierlichen Produktion nicht im Wege stehen, sind hier nötig [5].

4 Anwendungen

Wie in den Kapiteln 2 und 3 deutlich wurde, erreicht die Polymerelektronik aufgrund der schlechteren elektrischen Eigenschaften wesentlich geringere Leistungen als die anorganische Elektronik. Sie birgt hingegen dann Chancen, wenn durch die Verwendung neuer Substrate neue Anwendungen möglich werden oder die kontinuierliche Produktion die Massenherstellung einfacher Elektronik erlaubt. Bereits heute existieren erste Anwendungen die auf Polymerelektronik basieren, viele weitere zukünftige Anwendungen wurden erdacht. Um den Massenmarkt zu erobern ist aber weitere Materialforschung und die und Entwicklung und Optimierung der Herstellungsprozesse nötig.

4.1 Organische Leuchtdioden

Organische Leuchtdioden zählen zu den ersten Produkten mit erreichter oder annähernd erreichter Marktreife. Erklären lässt sich dies zum einen durch neue Anwendungsmöglichkeiten, aber auch durch die vergleichsweise geringen Anforderungen an die Produktionstechnik. So können schon heute OLED gedruckt werden [10].

Aufbau von OLED. Abbildung 7 zeigt schematisch den Aufbau einer organischen Leuchtdiode. Als organische Schichten können Polymere, aber auch sogenannte "small molecules" zum Einsatz kommen. Bei Verwendung einer Polymerschicht spricht man deswegen auch von PLED oder POLED [9]. Beim Anlegen einer

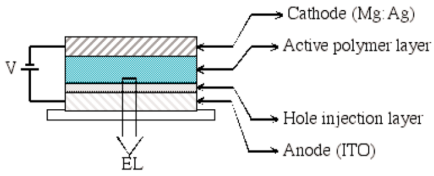


Abbildung 7. Schichtenaufbau einer PLED. Eine der Elektroden muss transparent sein, damit Licht austreten kann. Die Anode besteht zumeist aus Indium-Tin Oxid (ITO) [9]

Spannung werden sowohl Elektronen als auch Löcher in das aktive Material injiziert. Dort rekombinieren sie unter Anregung des Emittermaterials, und die Anregungsenergie wird als Photonen freigesetzt. Die Farbe bestimmt sich hierbei durch das Anregungslevel des verwendeten Emittermaterials [9].

Eigenschaften von OLED. Während anorganische Leuchtdioden Punktlichtquellen darstellen, so stellen organische Leuchtdioden eine flächige, diffuse Lichtquelle dar, da hier nicht nur an einer Übergangsschicht zwischen p- und n-Halbleitern Photonen erzeugt werden, sondern auf der gesamten Fläche des aktiven Materials [9]. Ausserdem verspricht man sich von OLED eine hohe Energieeffizienz, auch wenn diese bis heute noch in weiten Teilen nicht in der Praxis erreicht wird. Durch die Verwendung von Polymeren als aktives Material können OLED sehr dünn hergestellt werden (Dicke der organischen Schicht von ca. 100nm) [9]. Durch die Auswahl an Substraten sind auch völlig neue Anwendungen wie flexible Displays denkbar (s.u.). Chemische "Maßschneiderung" der Polymere ermöglicht es zudem prinzipiell jede Farbe herzustellen.
OLED erfordern nur eine geringe Versorgungsspannung, benötigen aber derzeit noch einen hohen Strom [9, 31]. Als weiterhin problematisch für die Herstellung marktreifer Anwendungen stellt sich die noch geringe Lebensdauer dar: OLED verlieren mit der Zeit sichtbar an Leuchtkraft. Während man für niedrige

Leuchtstärken bereits Lebensdauern erreicht, die erste marktreife Anwendungen ermöglichen, so ist die Lebensdauer für OLED mit höheren Leuchtstären noch zu gering.

Weiterhin erwartet man von der OLED Technologie durch kontinuierliche und großflächige Fertigung geringe Produktionskosten. Bereits heute können OLED gedruckt werden [10], im Allgemeinen gilt jedoch, dass die entsprechenden Produktionsverfahren noch entwickelt bzw. optimiert werden müssen um den angestrebten Kostenvorteil auch tatsächlich zu erreichen [9].

OLED als Leuchtmittel. Die Anwendung von (O)LED in Leuchtmitteln wird in der Zukunft vermutlich eine immer stärkere Rolle spielen. Hauptgrund dafür sind die (erwartete) Energieeffizienz und speziell bei OLED die durch Polymertechnik günstigeren Massenproduktionskosten [9]. Einen Vergleich der Effizienz verschiedener Leuchtmittel zeigt Abbildung 8. Mit OLED als diffusen Lichtquel-

Type of lighting	Lighting Conversion Efficiency (Lumens / Watt) Under Optimum Conditions (Driving Voltages, Current)
Incandescent	13 - 17 lm/W
Fluorescent	50 - 100 lm/W (typically 90 lm/W)
HID	50 - 130 lm/W
LED	30 - 50 lm/W
OLED	>50 lm/W (green) – **still at unacceptably high currents !**

Abbildung 8. Vergleich der Effizienz verschiedener Lichtquellen (Glühbirnen fallen in die Kategorie Incandescent) unter optimalen Bedingungen. Es wurden bereits OLED erreicht die mit Leuchtstoffröhren konkurrieren können, allerdings bisher eine zu hohe Stromaufnahme besitzen [9].

len sind Anwendungen möglich, die mit Punktlichtquellen nicht möglich sind: So können mit OLED zusammenhängende großflächige Beleuchtungen hergestellt werden. Zusammen mit der sehr niedrigen Bauhöhe sind damit leuchtende Tapeten, Decken oder gar Fensterglasbeschichtungen denkbar.

Überall wo großflächige diffuse Beleuchtung gewünscht ist könnten OLED als Leuchtmittel Einzug halten und Leuchtstoffröhren verdrängen. Dies dürften hauptsächlich öffentliche Gebäude, Fabriken und Werbebeleuchtung betreffen. Sollten effiziente OLED-Lichtquellen in Zukunft tatsächlich bei gleichen oder geringeren Kosten als Leuchstoffröhren verfügbar sein, könnte OLED-Technologie zu immensen Energieeinsparungen führen, da diffuses Licht einen Großteil des Gesamtmarktes ausmacht (durch Industrie und Gewerbe) [9].

Das bereits angesprochene Problem begrenzter Lebensdauer gerade bei hohen Leuchtstärken stellt jedoch eine hohe Hürde für Anwendungen der OLED Technik für Beleuchtung dar. Für großflächige Beleuchtungen mag dies jedoch mitunter nicht so stark ins Gewicht fallen, da durch großflächigere Abdeckung möglicherweise vergleichsweise geringere Leuchtstärken ausreichen, um die gewünschte Helligkeit zu erreichen [9].

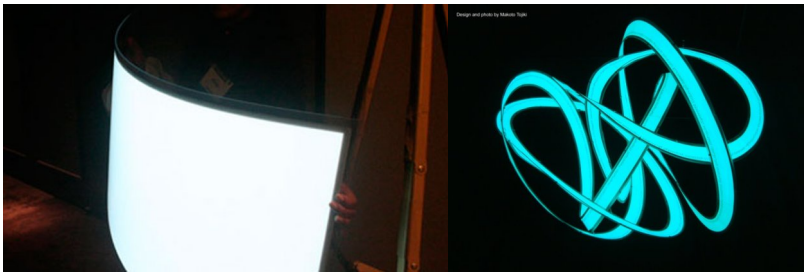


Abbildung 9. Links: Großflächige dünne und flexible Beleuchtung der Firma CeeLite. Hier kamen jedoch keine OLED sondern Lichtemitierende Kondensatoren zum Einsatz [11]. Solche Anwendungen sollen in Zukunft aber auch mit OLED realisiert werden. Rechts: Designerlampe "Archimedes' Dream" von Makoto Tojiki [12].

Neben großflächigen Leuchtmitteln erlaubt die Polymertechnologie auch die Verwendung flexibler Substrate. Diese ermöglichen eine Anzahl neuer Anwendungen. Flexibilität der Substrate, zusammen mit der sehr dünnen Fertigung von OLED, erlauben Anwendungen wie Leuchttapeten (ähnlich Abb. 9 links), aber auch neue Produktformen (Abb. 9 rechts).

Auch wenn erste Anwendungen mit OLED als Leuchtmittel schon vorgestellt werden konnten, so muss noch viel Forschungsarbeit geleistet werden. So müssen bspw. die ITO-Anoden (Abb. 7) für großflächige Anwendungen durch neue Materialien ersetzt werden, da diese den nötigen Stromfluss für solch große Flächen nicht gewährleisten können [9]. Weiterhin müssen Effizienz und Lebensdauer gesteigert werden. Gerade für die Erzeugung von weißem Licht aus verschiedenen Farben stört die stark unterschiedliche Lebensdauer der Farben rot, grün und blau. Hier führt die unterschiedlich schnelle Alterunggeschwindigkeit mit der Zeit zu Farbverschiebungen. Nicht zuletzt werden jedoch die erreichbaren Produktionskosten darüber entscheiden inwieweit OLED in den Beleuchtungsmarkt Einzug halten werden. Die Kosten sind bisher weitestgehend unbekannt (Abb. 10) [9].

OLED Displays. Stärker im Blickpunkt der Industrie als Leuchtmittel aus OLED sind OLED-Displays. Im Gegensatz zu LC-Displays benötigen OLED-Displays keine zusätzliche Hintergrundbeleuchtung, da OLED selbst Licht erzeugen und nicht nur filtern. Das Fehlen der Hintergrundbeleuchtung macht es möglich, sehr dünne Displays herzustellen. Die Selbstleuchteigenschaft von OLED-Displays sorgt zudem für sehr gute Kontrastwerte. Die flächenhafte Lichtabstrahlung von OLED im Gegensatz zu anorganischen LED (ILED) verhilft OLED-Displays weiterhin zu sehr guten Sichtwinkeleigenschaften. Prinzipiell sollten OLED auch weit energieeffizienterer arbeiten als bisherige Displays. Ein Problem ist auch hier jedoch die geringe Lebensdauer organischer Komponenten. Neben den grundsätzlichen Problemen mit Defekten durch Umgebungsfeuchte und Interaktion mit umgebenden Gasen sorgt auch hier das unterschiedliche Alterungsverhalten der Farben Rot, Grün und Blau für sichtbare Farbverschiebungen mit der Zeit. Als

	Incandescent bulb	Fluorescent tube	Fluorescent screw base	LED white	OLED white
Wall Plug Power (Watts)	75	20	20	0.072	0.08-0.18
Cost ($)	0.65	4.75	12.75	0.60	N/A
Lifetime, hrs	750	10,000	10,000	100,000	>30,000
Peak Efficiency, lm/W	17	60	60	100 (orange)	>50 (green)
Init. cost per (c)	0.05	0.4	1.06	42	N/A
Init. Cost per 1000 lm -hrs (c) *	0.07	0.04	0.11	0.42	N/A
Cost of Electricity per 1000 lm -hrs **	0.71	0.20	0.20	0.60	N/A
Total Cost per 1000 lm -hrs (c)	0.78	0.24	0.31	1.02	N/A

* Calculated using lifetime
** Calculated using $0.12 per kWhr
*** 0.08 for POLEDs, 0.18 for "Small molecular" devices. Due to rapid progress, these numbers may be already obsolete

Abbildung 10. Gegenüberstellung der Kosten verschiedener Leuchtmittel. Wo OLED-Leuchtmittel hier einzuordnen sind, ist noch nicht entschieden [9].

Anwendung von OLED-Displays hat die Industrie momentan vor Allem ultraflache und energiesparsame Fernseher, jedoch erreicht man bisher nicht die Bilddiagonalen von LCD Fernsehern.
Einen ersten solchen OLED-Fernseher brachte Sony mit dem XEL-1 November 2007 auf den Markt (Abb. 11). Dieser kam für umgerechnet 1200 Euro ausschließlich auf den japanischen Markt und wird nur in geringer Stückzahl (1300 Stck/ Monat) aufgelegt. Von einer wirklichen Marktreife von OLED-TV ist also nicht zu sprechen, das Modell ist eher als ein Proof-of-Concept zu werten. Trotz der vergleichsweise geringen Bilddiagonalen wurde die erste Charge innerhalb eines einzigen Tages verkauft [16]. Bis die Probleme bzgl. Lebensdauer und Produkti-

Abbildung 11. Bild links: Sony XEL-1 OLED-TV von der Seite betrachet. Daten: 11 Zoll OLED-Display, 3mm (!) dick, Kontrastverhältnis 1,000,000:1 (!), 960 x 450 Auflösung, 45W Leistungsaufnahme, Lebensdauer 30.000h [14],[13]; rechts: Auffallend ist das gute Bild trotz sehr flachem Sichtwinkel [15].

onstechnik (Massenproduktion) gelöst sind, werden OLED-Displays vornehmlich dort eingesetzt werden, wo die Anforderungen bzgl. Lebensdauer gering sind [7], also beispielsweise als kleine oder sekundäre Displays in Mobiltelefonen (Nokia 7900-Prism, Samsung E950 uvm.) [17] oder MP3-Playern (Sony ESeries uvm.) [17].

Neben den bisher betrachteten starren Displays können mit polymeren Materialien Displays prinzipiell auch auf flexible Substrate aufgebracht werden und somit biegsame Displays erzeugt werden. So wird an sog. "ePaper", also dem elektronischen Papierersatz, geforscht. Auch in Kleidung (wearable Computers) oder in Leuchttapeten (s.o.) könnte ein solches flexibles Display integriert werden. Doch auch hier steht die Forschung noch nicht vor der Marktreife. Das Hauptproblem organischer Displays, die geringe Lebensdauer, wird durch die Verwendung flexibler Substrate noch verstärkt: Können starre Displays noch in Glasschichten gekapselt werden, so muss bei flexiblen Displays auch diese Aufgabe ein Kunstoff übernehmen. Für OLED-Displays müssen sehr undurchlässige Barriereschichten verwendet werden, welche zudem farbneutral sein müssen. Die Verwendung anorganischer Materialien als Barriereschichten schränkt die Flexibilität der Displays ein und kann beim Biegen zu Bruchstellen in den Displays führen [18].

Abbildung 12. Links: Flexibles OLED Display von Sony [19], rechts: Headmounted Display LE-500 der Firma Liteye [21]

Neben flexiblen Displays wird durch Polymertechnologie eine weitere Sorte Displays möglich: Transparente Displays. Hier müssen alle Komponenten lichtdurchlässig ausgeführt sein. Auf dem Markt existiert bereits ein sog. "Head Mounted Display" mit transparentem OLED-Display [21]. Mögliche Anwendungen sind hier eher im militärischem Bereich zu sehen. Im zivilen Sektor könnte ein solches Display zukünftig aber auch als Assistenzsystem bei medizinischen Eingriffen oder zur Einblendung von Informationen in eine Winschutzscheibe dienen.

4.2 Solarzellen

Auch wenn momentan die OLED-Technologie bei den Anwendungen leitender Polymere im Fokus der Industrie steht, sind auch noch andere Anwendungen von Interesse.
Eine davon ist die Entwicklung neuartiger Solarzellen. Durch (Halb-)leitende Polymere ist es auch hier möglich von bisher verwendeten anorganischen Materialen (kristallines bzw. amorphes Silizium) auf flexible Substrate umzusteigen. Flexible Solarzellen werden hiermit prinzipiell möglich. Sie könnten in "Intelligente Kleidung" (s.o.) integriert werden oder die Spannungsversorgung anderer flexibler Anwendungen übernehmen. Auch hier erwartet man sich von der Polymertechnologie den möglichen Einsatz massentauglicher Produktionsverfahren und damit geringe Kosten.
In [24] wird eine Solarzelle auf Basis konjugierter Polymere mit einem Wirkungsgrad von 3 % beschrieben (wirtschaftlich interessant ab 5 %). Wird hierbei noch ein Glassubstrat verwendet, gehen die Autoren davon aus, dass diese Technik prinzipiell auch an flexible Substrate angepasst werden und evtl. in Druckverfahren hergestellt werden kann.

4.3 Sensorik

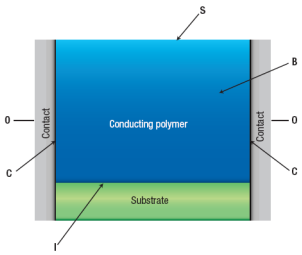


Abbildung 13. Einfacher organischer Sensor: Leitfähiges Polymer zwischen zwei Elektroden [6].

Während man die offene Struktur der Polymere und die damit verbundene Interaktion mit der Luftfeuchte und Gasen aus der Umgebung aufgrund der Lebensdauerverkürzungen und Veränderungen in den elektrischen Eigenschaften (s.o.) normalerweise als Nachteil sieht, so kann man diese Effekte jedoch auch gezielt für sensorische Anwendungen nutzen. So wurden u.a. auf Polymeren basierende Sensoren für die Messung von Gaskonzentrationen, Alkoholgehalt, Feuchtigkeit, PH-Werten [2] vorgestellt aber auch Photosensoren [25]. Dabei befindet sich ein Polymer entweder als Leiter zwischen zwei Elektroden (Abb. 13) oder als Kanal- oder Gatematerial in einem OFET (Abb. 14) [6]. Eindringende

Gase und Feuchtigkeit verändern hier die elektrischen Eigenschaften der Polymere. Eine Messung erfolgt dann durch Messen der Leitfähigkeitsänderungen, der Elektrodenpotenziale, Stromflussmessungen bei fester Spannung oder durch Stromänderungen bei Änderung der Spannung.
Im Allgemeinen sind solch gemessenen Werte aber schwierig zu interpretieren, da man nicht sicher sagen kann was genau zwischen den Elektroden passiert [6]. So verfälschen Übergangseffekte zwischen Polymer, Substrat und Elektroden die Ergebnisse, aber auch die unbestimmte Zeitspanne, die es dauert bis ein Polymersensor überhaupt von dem zu messenden Gas oder der Flüssigkeit durchdrungen ist, macht eine Auswertung schwierig. Auch hängen Sensorergebnisse mitunter zu einem großen Teil von der Präparation des einzelnen Sensors ab. Deswegen empfiehlt es sich anstelle einzelner Sensoren Arrays mehrerer Sensoren zu verwenden, deren (abweichende) Ergebnisse dann durch informationstechnische Methoden ausgewertet werden können. [23].
Neben der natürlichen Eignung von Polymeren für Sensorapplikationen spie-

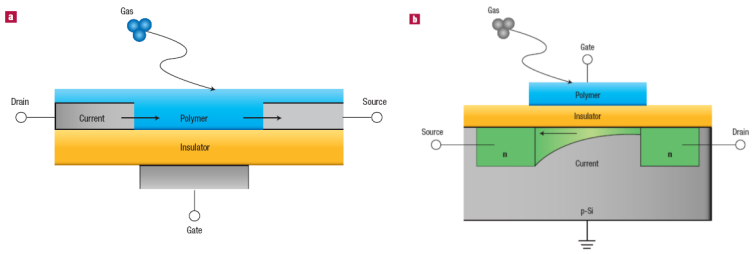


Abbildung 14. OFET als Gassensor [6].

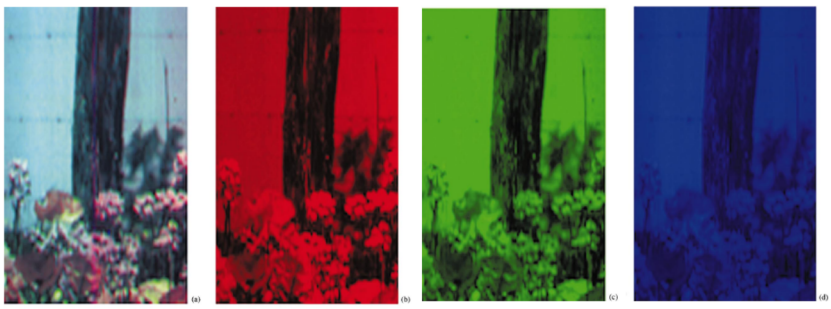


Abbildung 15. Aus den Einzelaufnahmen zusammengesetztes RGB Bild. 256^3 (= 16777216) Farbwerte konnten unterschieden werden.

len auch hier wieder die potentiell möglichen Massenproduktionsmethoden mit

den entsprechenden Kostenvorteilen eine wichtige Rolle. Die Verwendung neuer Substrate kann auch hier die Granularität der sensorischen Abdeckung hin zu großflächiger Sensorik verschieben. So wurde in [25] ein großflächiger Bildsensor auf Polymerbasis vorgestellt. Gezeigt wurde hier die Produktion photosensitiver Filme mit einer Fläche von bis zu 15 cm x 15 cm. Weiterhin wurden lineare Arrays von Fotodioden aus 102 Elementen hergestellt. Damit wurde zeilenweise ein aufprojeziertes Bild analog eingelesen und A/D gewandelt. Jedes Element hatte hierbei eine Auflösung von 40 dpi und das Array eine Gesamtlänge von knapp 65 cm. Ein RGB Farbbild wurde dann aus der Aufnahme dreier Graustufenbilder mit je 256 Farbwerten unter Verwendung von Farbfiltern erzeugt. Das Material erlaubt unter Verwendung breiterer A/D Wandler jedoch auch noch feingranularere Farbabstufungen. Abbildung 15 zeigt das zusammengesetzte Bild neben den Einzelaufnahmen.

4.4 Mensch-Maschine Schnittstellen

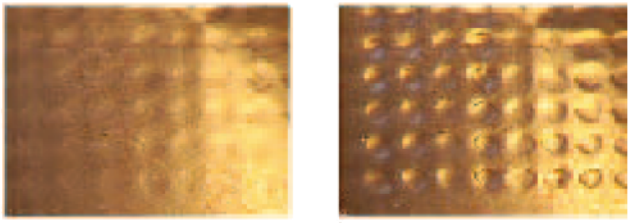


Abbildung 16. Braille-Display aus elektroaktiven Polymeren (EAP) [26].

Unter Ausnutzung piezoelektrischer Effekte (Formveränderung durch anlegen einer Spannung an das Polymer oder umgekehrt Erzeugung einer Spannung durch Verformung des Polymers) können Polymere auch neue Möglichkeiten in der Mensch-Maschine Interaktion eröffnen (sog. "elektroaktive Polymere", kurz EAP). So können beispielsweise Braille-Displays (Blindenschrift) aus EAP hergestellt werden (Abb. 16). Hierbei wird Schrift (z.B.: schriftliche Computerausgaben) in Erhebungen und Vertiefungen im Polymermaterial umgesetzt [26] um Sehbehinderten einen Zugang hierzu zu ermöglichen. In Zukunft sind solche taktilen Schnittstellen jedoch auch für den allgemeinen Gebrauch bei der Integration von Elektronik in dünne oder flexible Materialien denkbar.

4.5 Integrierte Schaltungen

Zusammen mit den oben vorgestellten OFET und Durchkontaktierungen (Via-Holes) sind die Herstellungen weiterer Logikgatter möglich. So wurden bisher erfolgreich Inverter und RS-Flipflops [27] produziert.
Für die Herstellung elektronischer Schaltungen ist das Vorhandensein eines Taktgebers von Notwendigkeit. Bestehend aus Invertern konnte bereits erfolgreich ein Ringoszillator im Druckverfahren hergestellt werden [3]. Die mit Polymeren erreichbaren Taktgeschwindigkeiten hängen zum einen von den Kanallängen der OFET, aber weiterhin auch stark von der Ladungsträgermobilität des Polymermaterials (s.o.) ab. Der in [3] erzeugte Ringoszillator erreicht eine Taktfrequenz von nur 1 Herz, Inverter konnten aber per Tintenstrahldruck schon mit Frequenzen von einigen hundert Kiloherz Schaltgeschwindigkeiten [5] gedruckt werden.

Auch eine komplett in Polymeren realisierte Schaltung konnte bereits gezeigt werden. So beschreibt [27] einen 4 Bit Parallel-Seriell Wandler mit Taktfrequenzen um 200 Hz.

RFID

Um das in der Einleitung erwähnte Szenario vom Ersatz bisheriger Barcodes auf Produkten durch intelligente Preisschilder (Funkettiketten) zu verwirklichen müssen diese massenhaft sehr günstig produzierbar sein. RFID Tags erfreuen sich für solche Anwendungen wachsender Beliebtheit.
(Passive) RFID-Etiketten bestehen im Wesentlichen aus einem kleinen Mikrochip, einem Kondensator und einer Antenne. Befindet sich der RFID-Transponder in der Nähe eines RFID Senders so wird über eine elektromagnetische Welle "continuous wave" der Kondensator geladen und die Spannungsversorgung des Mikrochips und die Energie für die Funkantwort des Etiketts damit sichergestellt. Die Informationsübertragung erfolgt dann, indem das Lesegerät die Abschwächung der Funkwellen durch den Transponder misst. Hochfrequenz-RFID Tags arbeiten hier beispielsweise mit der Frequenz von 13,56 Mhz. Die Ladungsträgermobilitäten bisheriger Polymere stellen eine sehr hohe Hürde für den Druck entsprechender Antennen mit diesen Frequenzen aus Polymermaterialen dar. Zwar werden schon heute RFID-Antennen industriell gedruckt, statt Polymertinten kommen hierbei aber mit (Silber-)Partikeln versetzte Tinten zum Einsatz. Für einen Masseneinsatz wie den Barcodeersatz ist diese Methode jedoch zu teuer [28]. Zwar berichten Phillips (Februar 2006) und PolyIC (2005) von der erfolgreichen Herstellung von RFID-Etiketten aus Polymeren. Eine Tauglichkeit für Massenproduktion ist hierbei laut [3] jedoch nicht gegeben. Dennoch fand bereits erfolgreich ein Feldversuch mit gedruckten polymeren RFID-Etiketten als Eintrittskarten auf 13,56 Mhz Basis statt [29].

5 Ausblick

Auch wenn bereits erste Anwendungen mit (Halb-) leitenden Polymeren gezeigt werden konnten, ist doch viel Grundlagenforschung auf dem Gebiet der Poly-

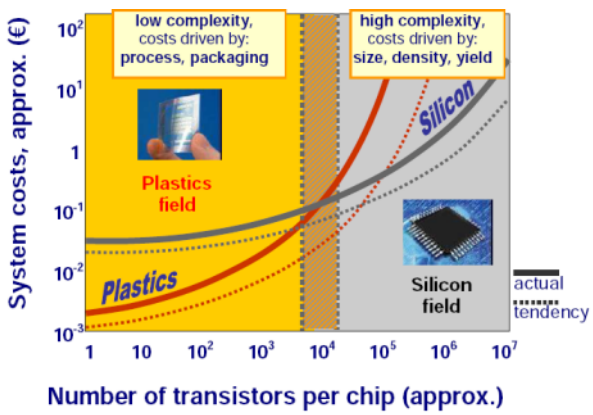


Abbildung 17. Vergleich anorganischer vs. organischer Elektronik und deren Kosten heute und in der Zukunft [1]

merelektronik nötig. Dies betrifft nicht nur die Materialien selbst, sondern auch die Produktionsmethoden. Um von den potentiell positiven Eigenschaften wie Energieeffizienz und niedrigen Produktionskosten profitieren zu können, werden noch Jahre an Forschungsarbeit investiert werden müssen.
So zeigen beispielsweise die Roadmaps von Toshiba [31] und Samsung [30], dass trotz erster Anwendungen von OLED-Displays am Markt noch viel Ausdauer nötig ist, um die hochgesteckten Ziele bzgl. Lebensdauer und Effizienz zu erreichen. Dennoch zeigt sich schon jetzt, dass Polymerelektronik leistungsmäßig nicht mit anorganischen Computern wird konkurrieren können, sondern vielmehr dann eingesetzt werden soll, wenn Elektronik zwar leistungsschwächer sein darf aber dafür in enorm hohen Stückzahlen verfügbar sein muss. Dies könnte in der Zukunft gerade auch bei Ubiquitärem Computing (Ubicomp) der Fall sein. Die hierbei angestrebte Integration von Elektronik in den Lebensalltag erfordert sowohl hohe Stückzahlen bei niedrigen Kosten, als auch neuartige Materialien und Sensorik. Diesen Anforderungen könnte Polymerelektronik in Zukunft möglicherweise gerecht werden.

Literatur

1. Alessandro Manuelli: “Influences of Printing Technologies on the Electrical Performances of Conjugated Polymers for Organic Transistors.” *Dissertation an der Fakultät für Maschinenbau der Technischen Universität Chemnitz*, 2006.
2. Basudam Adhikari, Sarmishta Majumdar:“ Polymers in sensor applications.” *Progress in Polymer Science Volume 29, Issue 7*, 2004 , pp 699–766.
3. M Bartzsch, U. Fuegmann, T. Fischer, u. Hahn, H. Kempa, K. Preissler , G. Schmidt, A. Huebler: “All-printed electronics and its applications: a status report.”. *IS&T, The Society for Imaging Science and Technology. DF 2006, International Conference on Digital Fabrications Technology. Denver, USA.* 17.09. - 21.09.2006, Proc. S. 13-16.

4. Sean A. Stauth and Babak A. Parviz: “Self-assembled single-crystal silicon circuits on plastic.” *Proceedings of the National Academy of Sciences of the United States of America*, September 19, 2006 vol. 103, no. 38, 13922-13927.
5. H. Sirringhaus, T. Kawase, R. H. Friend, T. Shimoda, M. Inbasekaran W. Wu, E. P. Woo: “High-Resolution Inkjet Printing of All-Polymer Transistor Circuits.” *Science 15 December 2000 Vol. 290. no. 5499*, pp. 2123–2126.
6. Jiri Janata, Mira Josowicz: “Conducting polymers in electronicchemical sensors.” *Nature Materials Vol 2* January 2003.
7. “Emerging Markets for Printed and OLED Lighting.” Market Research Report, NanoMarkets LC 2007.
8. Bundesministerium für Bildung und Forschung der Bundesrepublik Deutschland: Pressemitteilung 154, 2006. http://www.bmbf.de/_media/press/pm_20060911-154.pdf.
9. “Organic Light Emitting Diodes (OLEDS) For General Illumination”. An OIDA Report March 2001. Optoelectronics Industry Development Associaton, 1133 Connecticut Avenue, NW, Suite 600 Washington, DC 20036.
10. OLLA Project: Pressemitteilung 3/2007. www.olla-project.org, 14.5.2007.
11. Wilson Rothman: “CeeLite Flat Lighting Panels Are OLED for Giants.” http://gizmodo.com/gadgets/bendy-displays/ceelite-flat-lighting-panels-are-oled-for-giants-331023.php, 6.12.2007.
12. Makoto Tojiki: Archimedes’ Dream. http://www.makototojiki.com/download_center.html#archimedes.
13. Sony Japan: http://www.sony.jp.
14. Pro Physik: “OLED-Fernseher serienreif.” http://www.pro-physik.de/Phy/leadArticle.do?laid=9681.
15. Riyad Emeran: “Sony OLED Even Better In The Flesh October 2007.” http://www.trustedreviews.com/tvs/news/2007/10/02/Sony-OLED-Even-Better-In-The-Flesh/p1.
16. Plastics Information Europe: “OLEDs: First Sony televisions sold out in Japan.” http://www.plasteurope.com/pie-ticker/detail.asp?id=209696 Plastics Information Europe, Bad Homburg, 1/2007.
17. Oled-Info: Oled Devices. http://www.oled-info.com/devices.
18. Sugimoto, A.; Ochi, H.; Fujimura, S.; Yoshida, A.; Miyadera, T.; Tsuchida, M.: “Flexible OLED displays using plastic substrates.” *Selected Topics in Quantum Electronics, IEEE Journal of Volume 10, Issue 1*, Jan.-Feb. 2004, 107–114. DOI: 10.1109/JSTQE.2004.824112.
19. Golem.de: ‘Biegsames OLED-Display von Sony. RubrikHardware 29.05.2007.
20. Michael Kiy: “Organische Elektronik, Chips aus Plastik.” *Physik in unserer Zeit, Vol. 34, Issue 1*, pp.27–31. 2003.
21. Liteye.com, LE-500 Specifactions. http://www.liteye.com/Portals/0/LE500spec.pdf.
22. Liming Dai, Prabhu Soundarrajan, Taehyung Kim: “Sensors and sensor arrays based on conjugated polymers and carbon nanotubes.” *Pure Appl. Chem., Vol. 74, No. 9*, pp. 1753–1772 2002.
23. Michael S. Freund; Nathan S. Lewis: “A Chemically Diverse Conducting Polymer-Based ’Electronic Nose’.” *Proceedings of the National Academy of Sciences of the United States of America, Vol. 92, No. 7.* (Mar. 28, 1995), pp. 2652–2656.
24. Jürgen Parisi, Vladimir Dyakonov, Carsten Deibel und Ingo Riedel:” Perspektiven der Photovoltaik.“ *Einblicke Nr. 40*, Herbst 2004 Carl von Ossietzky Universität Oldenburg.

25. Gang Yu, Jian Wang, Jon McElvain, Alan J. Heeger:" Large-Area, Full-Color Image Sensors Made with Semiconducting Polymers." *Advanced Materials Volume 10, Issue 17*, pp 1431–1434, 1999.
26. Helmut F. Schlaak, Peter Lotz, Marc Matysek: " Muskeln unter Hochspannung - Antriebe mit elektroaktiven Polymeren". *Thema Forschung 2/2006*, pp 86–73. http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/37531/1/05-0308.pdf.
27. Jürgen Krumm, Elke Eckert, Wolfram H. Glauert, Member, IEEE, Andreas Ullmann, Walter Fix, Wolfgang Clemens: "A Polymer Transistor Circuit Using PDHTT." *Electron Device Letters, IEEE Volume 25, Issue 6 June 2004*, pp 399–401, DOI: 10.1109/LED.2004.829669.
28. "Das Gedächtnis der Dinge. Kunstoffe für die RFID Technologie." *Research. Das Bayer Forschungsmagazin Vol. 19*, p. 94.
29. Prisma-Projekt: Pressemitteilung 25.9.2007, PRISMA 200709-001, www.prisma-projekt.de.
30. Masao Oonishi, Nikkei Microdevices: [FPD International] "Samsung SDI VP Indicates OLED Panel Roadmap in Keynote Session." TechOn, http://techon.nikkeibp.co.jp/english/NEWS_EN/20071029/141477/, 29.10.2007.
31. Takuya Otani, Nikkei Electronics: "Toshiba to Release OLED TV Only After 'Power Consumption Less Than LCDs'." TechOn, http://techon.nikkeibp.co.jp/english/NEWS_EN/20071213/144198/, 13.12.2007.

(Alle Internetadressen und deren Inhalte vom Stand 4.01.2008.)

Seminar Ubiquitous Systems

Manageability of Wireless Sensor Networks

Abraham Taherivand

Abraham.Taherivand@stud.uni-karlsruhe.de

Supervisor: Luciana Moreira Sa de Souza

Karlsruhe, January 2008

1 Abstract

Wireless sensor networks (WSNs) are becoming an increasingly important technology, that will be used in a variety of applications such as environmental monitoring, infrastructure management, public safety, medical, home and office security, transportation, and military [1]. Cheap, massively deployed sensor networks are subject to frequent changes in the network topology caused by failures, node mobility, low link quality, and numerous other factors that influence the behaviour of such systems. In this work we analyze existing tools for fault management in wired networks and evaluate its applicability in WSNs.

2 Introduction

Wireless sensor networks are becoming increasingly attractive for enterprise scenarios since the improve on the concept of RFIDs. In many cases failures and malfunctions can lead to break-down states that should be avoided in such environments. Maintenance and failure management cost can easily dominate the overall costs. Without appropriate tools, costs associated to maintenance can become predominant in a business process. Providing maintenance in such networks can become quite challenging if the causes of malfunctions are not properly diagnosed specially in scenarios where the scale of the network makes it impossible for a human to process all the information to identify the malfunctioning parts. In many aspects, wireless sensor networks have similar behaviours when compared to wired networks which imply that management tools developed for wired networks could be adapted to wireless sensor networks. The focus of this research is to analyse existing network management tools focusing on Fault Management. For wired networks and its applicability in wireless sensor networks. The instant seminar-work will compare approaches, systems and tools for network management in wired and wireless sensor networks. Introductive there will be a general overview about the functions and requirements in network management approaches and systems. The focus of the instant seminar-work will be on 'Fault Management' as main aspect and angle of vision, while comparing the solutions in wired and wireless sensor networks. This research will end with a conclusion, summary and forecast of similar requirements for several tools, systems and approaches in both worlds.

2.1 Network Management in Wired Networks

A. Claims [2] refers to network management as the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance, and provisioning of networked systems. Under the 'OSI model' [3] network management takes account of five key areas and aspects: *configuration management, fault management, performance management, accounting management*, and *security management*. The following functionalities are also part of network management: controlling, planning, allocating, deploying, coordinating, and monitoring the resources of a network, network planning, frequency allocation, predetermined traffic routing to support load balancing, cryptographic key distribution authorization. A large number of protocols exist to support wired network management. Common protocols include:

- SNMP [4]
- WBEM [5]
- CMIP [6]
- Transaction Language 1 [7]
- Java Management Extensions [8]

Beside the protocols there are several tools and applications for network management in wired networks. In chapter 4 we present and discuss the most relevant tools, approaches and techniques in the field of wired network management.

2.2 Network Management in Wireless Sensor Networks

Wireless sensor networks (WSN) consist of many small sensors that are limited in resources, particularly processing power and battery life [9]. Generally network management is for any network a critical function, also for wireless sensor networks. The design of wireless sensor networks makes network management a more difficult task than that of traditional networks. There are several different key management aspects in WSN:

- Power Management
- Topology Management
- Fault Management
- Traffic/Performance Management
- Application Management
- Security Management

These aspects belong to the function of network management. In chapter 3 different approaches and tools will be shown to handle those key aspects in the context of network management in wireless sensor networks. WSNs are highly dynamic and prone to faults, mainly because of energy shortages, connectivity interruptions, and environmental obstacles. Network failures are common events rather than exceptional ones. Thus, in WSNs, tools and protocols are mainly concerned with monitoring and controlling node communication in order to optimize the efficiency of the network, ensure proper network operation, maintain the performance of the network, and control large numbers of nodes without human interaction [9].

3 Network Management Systems in WSNs

To understand the functions, aspects and difficulties of network management in WSNs, you have to determine the goal, intended use, design and architecture of the different techniques, approaches and applications. This chapter includes an overview of the existing techniques, approaches and applications for network management in WSNs. The most important and relevant approaches will be highlighted. However the focus of this instant seminar work is on *fault management*, so the following highlighted approaches will be analyzed against the background of that focus.

3.1 Organization, features and functionalities for network management in WSNs

A major difference of WSNs and wired network are the special features which are key factors on network management in wireless sensor networks. Special features of WSNs were discussed in [10]. They are:

- Different type of nodes
- Specific to the Application
- Resource Constrains
- Network Topology
- Fault Tolerance

These features affect the topic of network management in wireless sensor networks. The next subchapter will provide an overview and description about the existing approaches for network management in WSNs. The chosen system, respectively the different approaches which will be discussed in this seminar work, are strongly conditioned to their intended use respectively in which management level and functional area their inset are. The functional levels for WSN have been listed in chapter 2.2.

M. Yu, H. Mokhtar, M. Merabti define in [10] several architecture approaches. There are the 'Lightweight Management Architecture', 'Localized Management and Coordination', 'Generic Management Functions', 'Integration with Application-knowledge' and 'Adaptive Reconfiguration'. In wireless sensor networks, we come upon several protocols and architectures. In this subchapter the most important and common approaches will be introduced.

3.2 MANNA

One of the most famous approaches for management architectures in WSNs is called MANNA. A main aspect for architecture approaches in the context of wireless sensor networks is to define principles for the management architecture [11]. In their work they considered the following principles:

- Try to resolve problems derived from the dependencies that WSNs have on applications and energy restrictions

– Create a management function list, which include different functional areas, network functionalities and separate management levels.
– The generic configuration of a WSN has to be provided from a functional architecture
– Existing protocols, algorithms and mechanisms have to be adapted

To understand how MANNA works, we also have to take a look on the management service components and management functions. A better understanding of the interaction of 'WSN functionalities', 'Management Levels' and 'Functional areas' gives the following figure.

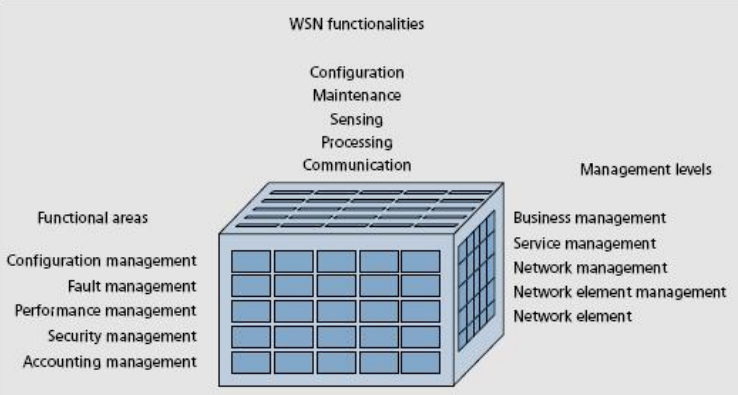


Fig. 1. : Three-dimensional organization for WSN management[1]

Figure 1 shows the three dimensions of 'WSN functionalities', 'Functional areas' and 'Management levels' and their relationship to each other. The new aspect in this three dimensional consideration are the WSN functionalities which are considered by the MANNA approach.

Management services are generally executed by a set of functions. For Ruiz, Nogueira and Loureiro [11] management functions represent the lowest granularity of functional portions of a management service, as perceived by users. The outcome of this is that the MANNA architecture determines that the wireless sensor networks management does not end in its functions, but rather policy management is supported. To adhere to the statement of Ruiz, Nogueira, Loureiro in [12], the MANNA architecture defines WSN models that represent aspects of the network and serves as a reference to the management functions. The relationship between WSN models will be defined in a Management Information Base (MIB). MANNA is adapting dynamicly the WSN behaviours by updating and analysing the MIB.

Some examples of WSN models were given in [13]. They are:

- Audit map
- Topology map
- Communication coverage area map
- Residual energy map
- Sensing coverage map

The Audit map gives an account of the security status of the nodes and possible hacking attempts. The topology map and communication map describes the presenting communication range of the nodes and their connectivity and thus the complete reach ability of the network. The residual energy map gives an overview of the battery level of the network nodes, and with the sensing coverage map is the area described which is covered by sensor elements.

Figure 2 depicts relationship between services, functions and WSN models.

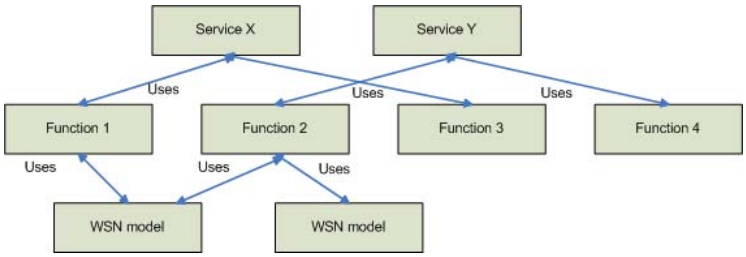


Fig. 2. : Relationship among services, functions and WSN models in MANNA management

The management functions, which consist of a service management, include three characteristics: Automatic, semi-automatic and manual. Further on the five different states of a function amount to: ready, not-ready, executing, done and failed. In [13] W. L. Lee et al. define possible management functions. They amount to coverage area supervision, network operating parameter configuration, topology map discovery, network connectivity discovery, aggregation discovery, energy map generation, and node localization discovery. The focus of Riuz et. al in [1], is to design a network management architecture with no specific MAC or routing protocols. In their research, they are suggesting an agent-based framework to distribute management functions to managed systems in which agents collect management information from the sensor nodes. Further on, in [14] it is explained, that MANNA provides automatic management services, which can be performed for fault management for event-driven applications. The aim of fault management is to detect failures of the network by analysing the WSN models. Two main management services are provided by MANNA: failure detection service and coverage area maintance service.

3.3 WinMS - a adaptive policy-based management system

The 'Wireless Sensor Network Management System' (WinMS) proposed by Louis Lee at al. in [9] is another approach for WSN management focused on fault management. There are some similarities with the MANNA approach, but WinMS got some different appendages. For an overview, figure 3 displays the WinMS architecture illustrating the relationship between management services, management functionalities and WSN models.

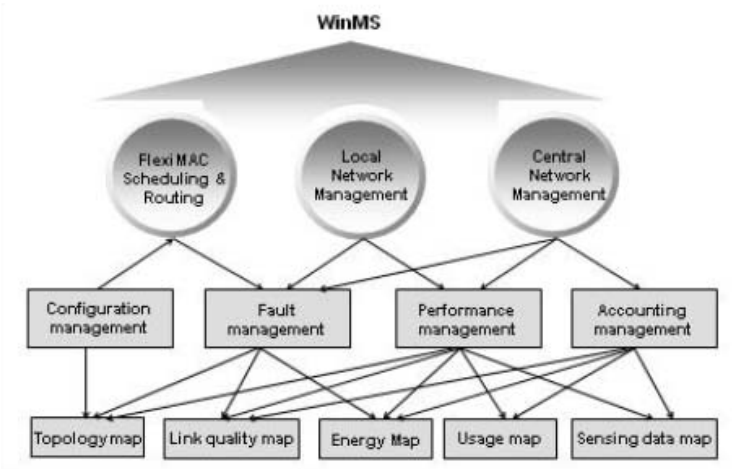


Fig. 3. : WinMS Architecture[9]

First management parameters with thresholds are predefined on the sensor nodes by the user. These are used as so-called event triggers. Further on the user specifies management tasks which are executed when the events are triggered. Yet another aspect of WinMS is, that there is an underlying routing protocol also used. This is called FlexiMAC. This protocol has the assignment to schedule the node communication as well as collecting distributed data from sensor nodes. Louis Lee et al. explain in [13] FlexiMAC as a TDMA-based protocol that provides synchronized communication using not a fix slot structure. Further on, all the nodes in the network build their own gathering schedules

The advantages of WinMS are:

- Energy efficient network management (TDMA-protocol)
- Local topology repair is warranted by the TDMA-protocol
- Self-configuration and self-stabilization with no human intervention

In [9] the authors propose WinMS as an adaptive policy-based network management system, "that continually observes network states and performs management functions to maintain the performance of the network and achieve effective networked node operations. WinMS provides a local and central management scheme that allow a network to be adaptive to changing network conditions

(e.g. topology changes, fluctuations in network behaviour, and event detections) by allowing it to reconfigure itself based on current events as well as predicting future events." WinMS propose a novel management function, which is called 'systematic resource transfer'. It provides automatic self-configuration and self-stabilization local and global. As described in [9], the novel management function enables a node's time slot to be used by other nodes in the network. In [13] the initial setup cost for building a data gathering tree and node schedule, which is proportional to the network density, is regarded as a disadvantage. But this argument could be tolerated in view of the advantages.

3.4 Sympathy – a Debugging System

"In wireless sensor networks, bugs are difficult to track because they are often multicausal, non-repeatable, timing-sensitive and have ephemeral triggers such as race conditions, decisions based on asynchronous changes in distributed state, or interactions with the physical environment" [15]. Failure detection and debugging is one of the main aspects in management of WSNs. In this subchapter the most common debugging system, called Sympathy [15], will be introduced. To understand how Sympathy works, we will present an overview of the architecture of this system. The system is designed for gathering data in wireless sensor networks. A centralized sink is collecting the data from the several nodes. The nodes transmit different readings to the sink in regular time intervals. The abandonment of the sink is, to analyze those metrics for identifying possible and potential failures.

On inferring an event, Sympathy [15]:

- Stores all metrics it has collected from the past 200 time units for the node that caused the trigger, providing temporal context.
- Stores all metrics it has collected from the past 200 time units for the nodes neighbouring the node where the event was detected, providing spatial context.
- Prints event and context information to a log file, which can aid in correlating events.
- Calls applications interested in the event

Above all, we have to take a look on the metrics. The sink collects the data from the metric in two ways. Scanning the channel and from regular transmission of metrics by nodes. The nodes send metrics to the sink every metric period. It has to be separate into three different types of metric. *Component flow*, *generic and connectivity*.

W. L. Lee, A. Datta, R. Cardell-Oliver [13] say that Sympathy has two types of nodes: A Sympathy-sink and a Sympathy-node. The Sympathy-sink receives requested data from the Sympathy-node. The Sympathy-node is a sensor node that collects and monitors network metrics. Figure 4 shows a general architecture of a simulated system running Sympathy.

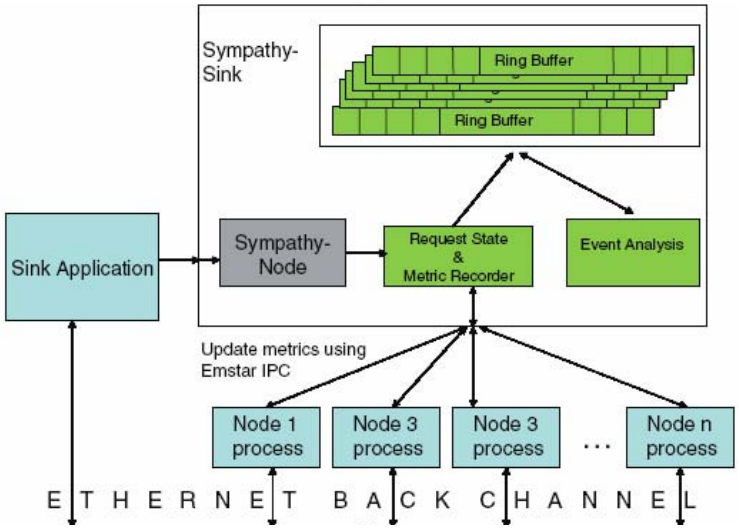


Fig. 4. : General architecture of a simulated system running with Sympathy [16]

N. Ramanathan et. al. [16] explain that the Sympathy-sink receives updated metrics from the Sympathy-node and processes it using the Emstar IPC framework. Emstar [17] is a software environment for developing and deploying wireless sensor networks. The Sympathy-sink contains certain components to record metric data from any node, process metrics to identify events, and record context.

The principle for the failure analysis in Sympathy is based on monitoring the traffic of the nodes. If the traffic is less than expected, a failure is detected and the localization is been triggered. The identification of failures are based on the reasons of bad path to sink, bad node transmit, bad path to node, no route to sink, no neighbours, node rebooted, node crashed [18]. Sympathy is one of the most important debugging systems for sensor networks. Since Sympathy handles the entire process of debugging and fault analysis, it differentiate itself from other existing tools. The advantages of Sympathy are [13]:

- Ability to highlight failures
- Localization of the cause of failure by correlating detected events with metrics to determine their spatio-temporal context.
- Events of interests can be proactively identified

The major drawbacks of Sympathy summarized can be summarized as:

- No automatic bug detection
- Historical data and post-mortem analysis of metrics.
- Require nodes to exchange neighbourhood lists. That means high energy consumption and delayed node communication

– Inaccurate analyses cause of imprecise global network knowledge of the Sympathy-sink

3.5 Self-Monitoring System

In [19] a self distributed self-monitoring mechanism is proposed. It is called 'Two-Phase Self-Monitoring Mechanism for Wireless Sensor Networks' (TP). It is assigned for monitoring applications in sensor networks. As [19] described, health monitoring of individual nodes is important for detecting malfunctioning nodes and intrusions which can result in the destruction of nodes. A two-phase timer scheme for local coordination and active probing is used by the TP system. In the first phase of the two-phase timer scheme, a sensor node waits for updates from a neighbour sensor node. Further on a node collaborates in the second phase with its neighbours to identify a condition.

Table 5 shows the differentiation of explicit and implicit fault detection in TP.

Fault detection implicit	Fault detection explicit
Discovering failed node communication (e.g. physical damage or energy depletion)	Nodes analyse sensor data, and will trigger an alarm if a relevant event arises (e.g. Temperature changes). Based on neighbourhood information's.

Fig. 5. : Three-dimensional organization for WSN management

For implicit fault detection, the sensor nodes must be monitored without interrupts. So, a distributed scheme is used for monitoring the activity of the sensor nodes. It is separated in active monitoring (implicit fault detection) and passive monitoring (explicit fault detection). In [13] the authors explain that each node sends 'alive' update messages to its neighbours during monitoring its neighbours sensors actively. For those 'alive'-messages a certain period of time (also called response time) is predefined. As it is explained also in [13], this scheme will fail if there is a network partition. Explicit fault detection is performed through neighborhood coordination and an alarm is sent to the base station only if the sensor nodes have high confidence that a fault has occurred. When a sensor node detects an event, the sensor node transposes information with its neighbour first, to explore the event correctness before releasing an alarm.

Beside the scheme approach for implicit and explicit fault detection, the TP system also got shortcomings. In [13] the authors mentioned as the main drawback, that the responsiveness of TP depends on the effectiveness of the routing protocol in propagating the alarm. Furthermore there is no synchronization

scheme implemented in TP, which means that timers are only updated during the packet reception.

3.6 Overview of design aspects and criteria for network management systems in wireless sensor networks

The following table gives an overview and summarise the previous elucidated relevant applications, systems and approaches for network management in WSNs with focus on fault management.

Network management system/application	Main management functionalities	Type	Robustness	Adaptability	Memory efficiency	Scalability	Energy efficiency
MANNA	Policy-based management framework, network state retrieval, sampling frequency control, coverage maintains and fault detection	Architecture / System	NA	NA	NA	NA	NA
Sympathy	Fault debugging	Debugging-System	Yes	Yes	Yes	No	Yes
Two-Phase Monitoring System	Local fault detection schemes	Monitoring	Yes	Yes	No	Yes	Yes
WinMS	Network state retrieval, synchronization, local repair, and systematic resource transfer	Architecture and Management System	Yes	Yes	Yes	Yes	Yes

Fig. 6. : Three-dimensional organization for WSN management[13]

Fault detection is one key focus of the systems 'MANNA', 'WinMS', 'TP' and 'Sympathy'. As it is said in [13], MANNA provides fault detection based on analysis of gathered WSN data. The WinMS approach schedules a period of time, in which nodes are listening to neighbours activities. They can self-configure themselves, if there is an event of failure, without knowledge of the hole network topology. "In addition, WinMS provides a centralised fault management scheme that analyses network states to detect and to predict potential failures and takes

corrective and preventive actions accordingly" [13]. In contrast, no automatic interaction and reconfiguration are provided by 'MANNA', 'Sympathy' and 'TP'. These tools focus on debugging and fault detection. The local fault detection in 'TP' is provided by monitoring the health of the nodes (its own and of the neighbour nodes). The debugging technique which is provided by Sympathy, localize faults which might appear from interaction of multiple nodes.

4 Wired network management

Network management is the management, operation and monitoring technology of IT networks and telecommunications networks. 'FCAPS' is the short cut for *Fault, Configuration, Accounting, Performance, Security,* which are the management categories [20]. The ISO model defines those network management tasks as:

- (F) Fault Management (identification, record, report and correct errors from occurring states).
- (C) Configuration Management (identification of all components (Configuration Items), which must be monitored).
- (A) Accounting Management (tracking the use of the network).
- (P) Performance Management (market value for money collecting data and statistics).
- (S) Security Management (authentication of users, authorizing access and use).

The comprehensive management of an organization's information technology (IT) infrastructure is a fundamental requirement. IP networks, especially wired networks, are often using Simple Network Management Protocol (SNMP) for the network management approach. Another approach is for example the Windows Management Instrumentations (WMI). In larger networks, also other protocols such as 'SAA' or 'Netflow' are used. In the national and international telecommunication networks, proprietary protocols with higher capacity are used, in Germany, for example QD2, in north of America 'TL1'. Because in comparison with private networks significantly higher number of network elements are used. Therefore polling based protocols such as SNMP cannot be applied. In Europe also OSI-protocols are used. Elaborate on the idea of using the existing approaches in WSNs, a large number of protocols and approaches exist to support network and network device management. The most relevant and important protocols and approaches for this seminar work will be introduced in the following subchapters.

4.1 SNMP Protocol

The Simple Network Management Protocol (SNMP) forms part of the internet protocol suite as defined by the Internet Engineering Task Force (IETF).

SNMP is used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. It consists of a set of standards for network management, including an Application Layer protocol, a database schema, and a set of data objects. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications. The Simple Network Management Protocol, is a network protocol, which has been developed by the IETF, for network elements (e.g. router, server, switches, printer, computer etc.) to supervise and to steer from a central place. The protocol regulates the communication between the supervised devices and the monitoring station. For this SNMP describes the structure of the packets, which can be sent. SNMP was designed in such a way with that each networkable equipment can be one monitoring tasks. To the tasks belong:

- Monitoring of network components
- Remote-control and remote-configuration of network components
- Error-detection and error-notice

Figure 7 elucidates the functionality of the SNMP protocol.

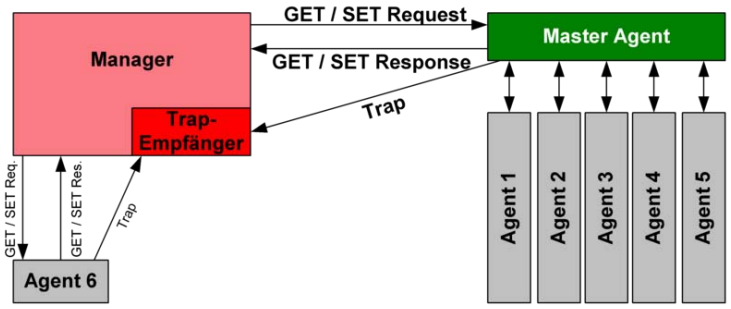


Fig. 7. : SNMP Functionality

For monitoring so called agents are used. It concerns programs, which run directly on the supervised devices. These programs are able to capture the condition of the equipment and make also separately attitudes or release actions. In SNMP, it is possible that the central management station with the agents can communicate over a network. In addition there are six different packets, which can be sent: The three GET packages (GET, GET-NEXT, GetBulk) can be sent from the manager to an agent, in order to request data over the respective station. This answers with a response package. With the SET package a manager can change values at the agent. With this, it is possible to make attitudes or released actions. The agent confirms the assumption of the values with a response package. If the agent recognizes an error during the monitoring of the system,

it can announce these with the help of a trap package without being asked to the management station. The manager does not confirm these packages. The agent cannot determine whether the trap arrived with the manager. So that the network load remains small, for reaching a better connecting performance, the UDP protocol is used. The agent receives with it the inquiries (Requests) on the port 161, while for the manager the port 162 is prescribed for the receiving of the trap messages.

4.2 3com Supervisor

There are a lot and several different tools, systems and approaches for managing a wired network. With the background of possible applicability of those tools in wireless sensor networks, a proprietary approach will be introduced in this subchapter.

The 3com Supervisor is a frontend for the administration of 3com switches. Over that frontend there are the following functions possible as described in [21]:

- "Increases the number of IP devices supported by 3Com Network Supervisor from 1,500 to 2,500
- Offers option of bulk software agent upgrades of all 3Com devices or upgrades according to device type or other criteria
- Provides the capability to perform bulk backup and restore of 3Com switch configurations to maximize productivity
- Allows scheduling of key tasks (network discovery, agent upgrade, configuration backup) to run at regular intervals
- Enables viewing and generates reports of detailed VLAN information against devices and links when managing networks"

The following figure 8 displays the 3com supervisor web-frontend.

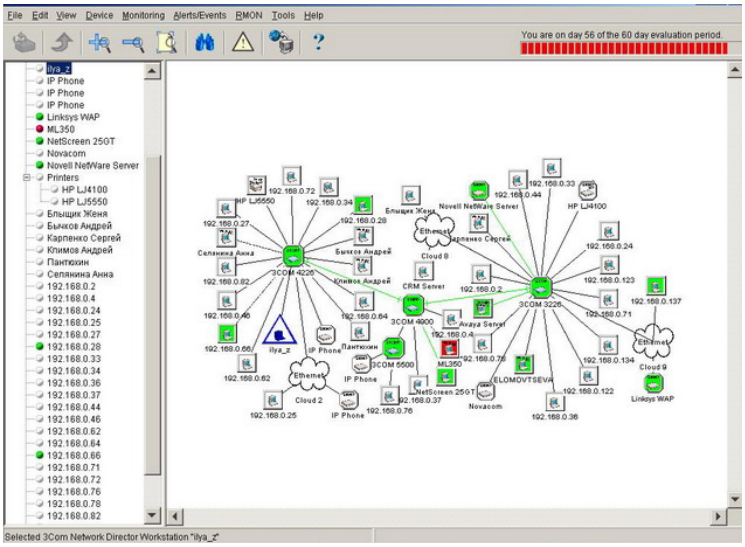


Fig. 8. : 3com Supervisor

The network-administrator has a full overview of the network state, including the servers, machines, workstations etc. Over this frontend it is possible to localize a possible server-problem which will shown if there is for example a problem with the connection. The network is shown as a network tree with the connections between the used hardware and the network-switches. Therefore it is possible to control the separate network nodes and the different 'local area networks' for example 'virtual local area networks' with other ip-adress-pools. There are no ping-packages sent to the availability of network components. It will be partly snmp-informations queried. In addition, a proprietary 3com-protocol is used for failure detection.

4.3 Nagios

Nagios is one of the famous and rampant monitoring solutions for wired networks. This open source computer system watches hosts and services in the network infrastructure, and sends alerts about the health status in predefined ways. In this subchapter, an overview about its functionalities and services will be elucidated. The following list gives an overview of the monitoring possibilities:

- Monitoring of several network services (e.g. SMTP, FTP, HTTP, SNMP etc.)
- Monitoring of environmental factors like temperature
- Monitoring of resources of several and different hosts in the network (e.g. CPU Usages, processor load, hard disk usage etc.)

A detailed feature overview is given in [22]. Figure 9 shows the detailed status of several services in a network which are monitored by Nagios.

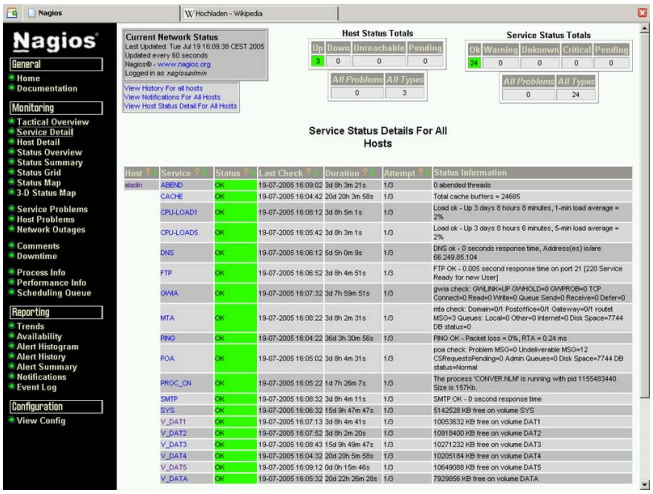


Fig. 9. : Monitoring services with Nagios

Figure 9 displays the service status for all the hosts in the network. If the status is highlighted green, there are no problems detected for the running and configured services of the network hosts. Nagios sends, so called 'keep alive' packages, to check if the host is available. For that, Nagios sends a network ping to the monitored hosts. If the host will not respond in the configured response time, there will be an alert, and the status will be marked as 'bad' status with red highlighting. The monitoring state for services could either be 'up', 'down', 'unreachable', 'ok', 'warning', 'critical', 'unknown', 'pending'. The checks Nagios will automatic do, are called 'active check' or 'passive check'. In [22] it is described that Nagios constantly needs to know the state of host or services. This process is called a check. Dondich also explain, when Nagios receives a change of monitoring status from a check, it initially declares the state as 'soft'. Nagios does not alert soft state because the check could have returned a failure, but the service could have recovered quickly, so it could have simply been a fluke. After the service has been in the same soft state after configured amount of checks, Nagios will change the monitoring state of the host or service into a 'hard state'. Figure 10 shows the Nagios frontend with an overview of the network infrastructure.

Figure 10 shows a circular status map with several information about the network hosts. The administrator is given a complete overview of the health state of the hosts, and he could localize possible problems in the network infrastructure. Nagios could also be configured in that way, that the network administrator will get direct remote-access to the hosts via the shown browser front end to do administrative work. Nagios is, because of its range of possibilities to configure, very powerful. It is possible to write plugins for special hosts, services or requirements in the network. The configuration is done over text configuration

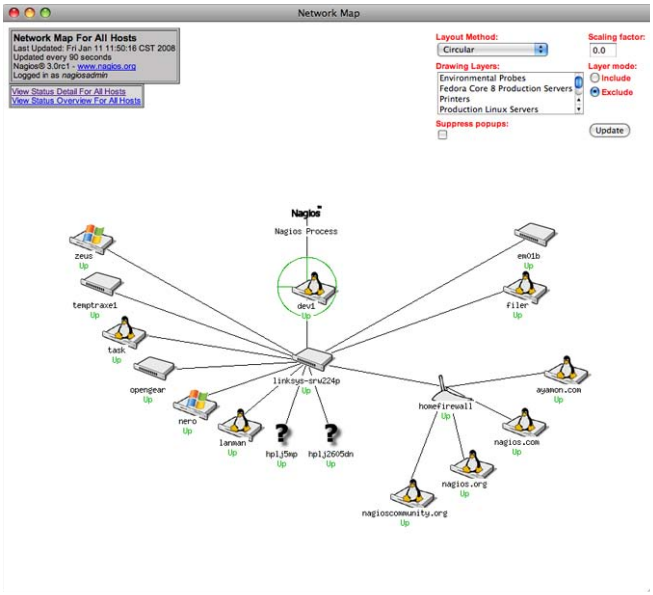


Fig. 10. : Circular Status Map in Nagios

file in an easy way. A main advantage is the scalability of Nagios and the varios possible configurations the system accepts. On the other hand the first configuration of Nagios take a long time and the built up of Nagios is very complex. In defiance of the disadvantages, for wired network monitoring solutions Nagios is one of the functional network monitoring and fault management systems. Only proprietary solutions like the IBM Tivoli (http://www.ibm.com), or the HP OpenView (http://www.hp.com) can reach the functionality and possibilities in network monitoring.

4.4 Overview of evaluated wired network management tools

The approaches presented in chapter 4 fulfill the requirements for fault management in wired networks. The main aspect are monitoring and centralized network administration. Nagios reach, in the field of monitoring all network administrative requirements. With its scalability and flexibility it is possible to have a real time health status overview of the hole network topology, services and hosts. SNMP was presented as the most famous protocol for monitoring and administration of wired networks. Its use is very common and practicable in wired networks, because most of the vendors produce their hardware devices (e.g. routers, switches, servers, etc.) with SNMP-support. An important aspect in wired networks is also, that the most vendors of network devices offer their own network management support tools. As an example for that, the '3Com Supervisor' was presented in this seminar-work.

5 Comparison

In [23] the author indicate that a concise and useful definition of Network Management can be found at the Cisco web site. It states: "Network management is a service that employs a variety of tools, applicatiobs and devices to assist human network managers in monitoring and maintaining networks." The FCAPS-model described in chapter 4 also represents the major aspects that must be considered when analyzing network management tools. Nevertheless there are several new functional areas of network management in sensor networks. These new functional areas introduced for network management of WSNs are topology management, energy management and program management[24]. The focus of the seminar work has been on fault management as functional area of network management.

The approaches presented in the previous chapters, show that there are several similarities from the requirements point of view when comparing wired and wireless networks. Nevertheless WNS, consist of many nodes, each with multiple links connecting to other nodes [25]. Information moves hop by hop along a route from the point of production to the point of use. In a wired network like the Internet, each router connects to a specific set of other routers, forming a routing graph. In WSNs, each node has a radio that provides a set of communication links to nearby nodes. By exchanging information, nodes can discover their neighbours and perform a distributed algorithm to determine how to route data according to the applications needs. Although physical placement primarily determines connectivity, variables such as obstructions, interference, environmental factors, antenna orientation, and mobility make determining connectivity a priori difficult. Other sustainable differences are that wireless sensor networks are ad hoc networks, which means networks without a fixed infrastructure between terminals. Ad-hoc networks are meshed networks, in which nodes with one or more neighbors are connected. This results in a multi-hop communications, in which messages from node to node are passed, until they have achieved their goal. Such networks are characterized by an unpredictable, dynamic behavior, because unlike permanently installed computer networks where the network topology are uncertain: the number and locations of the nodes, and 'line quality' are not predictable, while operating nodes can be added or fail without warning. In contrast, wired networks have a fixed infrastructure. Wired Networks are no ad hoc Networks and therefore have a different infrastructure and are thus to the other in the practice at other levels to manage.

In the research of this seminar-work, and as it is explained in the chapters below, several and different approaches, tools and systems for network management in wired and wireless sensor networks have been illustrated, with the focus of Fault Management as functional level in network management

"A main problem of network management in WSNs is, that we are mainly concerned with monitoring and controlling node communication in order to optimize the effiency of the network, ensure the network operates properly, maintain the performance of the network and control large numbers of nodes without human interaction" [9]. As described in [23], Network management in WSN

can include a wide collection of middleware services (e.g. distributed debugging, dynamic programmability platforms), distributed application (e.g. coverage monitoring, energy monitoring) and planing tools (e.g. deployment strategy). The open question is, whether there is a general model to enclose all WSN managing functions under the same roof. Certainly, a universal way of defining data/information related to management is needed (as in traditional networks). But Network Management in WSN is not retrieval of data from individual nodes (as in traditional networks where a general protocol can be defined for this purpose).

Comparing the presented tools, systems and approaches of both worlds, the functions and possibilities of them seems almost similar, even though fundamental differences previously discussed exist (e.g. in network topology, network devices, etc.). Also wired network management approaches are more field-tested and are developed for several years. The presented management tools and approaches for wireless sensor networks are under heavy development and not field-tested yet.

5.1 Conclusion

So, it is possible to apply the existing techniques from wired networks to wireless sensor networks? As the research of this seminar work shows, it is not possible to apply one-on-one the approaches, systems and tools from the wired networks to WSNs. Even if the base requirements to manage a network are the same. As we see, WSN have more requirements that are special and viewing ways than traditional wired networks. A good approach to the right direction for an integral network management solution could be MANNA, which was in the previous chapters shown. Because of the holistically idea and inclusion of the important and special viewing of functional levels it is the promising approach for a network management architecture and system for WSNs. As it is said in [1] until now, WSNs and their applications have been developed without considering a management solution. The MANNA architecture considers three managment dimensions: functional areas, management levels, and WSN functionalities. These dimensions are specified to the management of WSN and are the basis for a list of management functions. Until now there is no other approach which include this angle of view.

Another promising approach could be through the protocol level. In Tradtional wired networks the most of the network management software or tools are based on a special protocol. Most providers of network management solutions directed their solutions to existing standard protocols like the SNMP-protocol or over the IP-protocol. For WSNs there is a management framework called 'Sensor Network Management Protocol (sNMP)' proposed in [26]. As Lee et. al [1] describe, the sNMP framework has two functions. First, it defines sensor models that represent the current state of the network and defines various network management functions. Second, it provides algorithms and tools for retrieving network state through the execution of the network management functions. Models for sensors include network topology (node connectivity), energy map (node

battery power), and usage patterns. However, this approach is not as powerful and lasting as the SNMP-protocol in wired networks. Therefore it must have research and development in terms of simplicity, modularity and versatility in the context of network management in WSNs. Also in [1] it is said, that the development of general purpose network management layer protocols is still a challenging problem and remains a largely unexplored area for wireless sensor networks. An other open question is the development of management policies and expressive languages or metadata for representing management policies.

A comparison with the existing SNMP-protocol in wired networks is not readily available and workable. To reach the goals for a 'wireless sensor network management system' more continual research in many different management perspectives is needed. It is necessary and useful to explore the method of resolutions of wired network management solutions and approaches, but WNSs differ from wired networks, as it was explained in the chapters above. The challenge for network management seems to be similar in both worlds. However network management in general, is a very complex subject and it is even more complex for WSN, because of the differences concerning the hardware, software, architecture, communication protocols, specifications and requirements. For that reason, future studies should be inspired on existing solutions in wired networks. However, the specific requirements and circumstances of WSN must be considered.

To reach the goals for a network management system, more continual research in many different management perspectives are needed. It is necessary and useful to explore the method of resolutions of wired network management solutions and approaches. The challenges for network management are in both worlds nearly similar. At the time, the promising approaches for network management (with focus on Fault Management) in WSN's are 'MANNA' and 'WinMS' because of their wide range of possibilities in future development.

Merkmalsextraktion und -selektion

Helge Backhaus

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer: Martin Berchthold

Zusammenfassung Merkmalsextraktion und -selektion wird überall dort benötigt, wo auf Basis von durch Sensoren erfassten Daten ein Zustandserkenner erstellt werden soll. Nur selten können Umgebungs- und Benutzerzustände direkt aus Sensordaten gefolgert werden. Meistens ist eine Trennung von wichtigen und unwichtigen Merkmalen, oder das extrahieren neuer Merkmale aus bereits vorhandenen nötig. Die vorliegende Arbeit stellt einige Methoden zur Merkmalsextraktion und -selektion und einige praktische Anwendungen der Methoden vor.

1 Einleitung

Die Merkmalsextraktion und -selektion stellt einen wichtigen Zwischenschritt bei der Verarbeitung von Daten in ubiquitären Systemen dar. In der Regel möchte man durch Sensoren gewonnen Daten einen bestimmten Zustand zuordnen. Dies geschieht mit Hilfe von Zustandserkennern, die es einem ermöglichen sollen, anhand der erfassten Daten Rückschlüsse auf die Umgebung und den Benutzer von ubiquitären Systemen zu ziehen. Dabei kann eine Vielzahl von Sensoren über einen längeren Zeitraum zum Einsatz kommen, was häufig zu einer großen Menge von Daten führt. Jeder einzelne Sensor liefert dabei eine Datenreihe, welche oft auch als Merkmal bezeichnet wird. Bevor es jedoch möglich ist anhand dieser Merkmale Schlüsse über den Umgebungs- oder Benutzerzustand eines ubiquitären Systems zu ziehen, müssen die vorliegenden Daten in der Regel noch gefiltert und aufbereitet werden.

Für die Erstellung eines effizienten Zustandserkenners ist es notwendig, die Anzahl der zu betrachtenden Merkmale zu reduzieren. Mitunter kann direkt zwischen wichtigen und unwichtigen Merkmalen unterschieden werden und für den Zustandserkenner unwichtige Merkmale werden einfach aussortiert und ignoriert. In der Praxis ist die Unterscheidung zwischen wichtigen und unwichtigen Merkmalen jedoch kompliziert und die Anzahl an Möglichkeiten einige Merkmale aus vielen auszuwählen ist sehr groß. Zudem besteht noch die Möglichkeit neue, für den Zustandserkenner besser geeignete, Merkmale durch das Kombinieren bereits vorhandener Merkmale zu erhalten. In dem Fall gibt es sogar unendlich viele Möglichkeiten neue Merkmale zu extrahieren.

Die Aufgabe der Merkmalsextraktion und -selektion ist es daher, mittels geeigneter Verfahren aus einer gegebenen Anzahl an Merkmalen einen Satz Merkmale auszuwählen und zu extrahieren, die zu einem guten Zustandserkenner führen. Oftmals weisen direkt von Sensoren stammende Daten auch ein starkes

Rauschen auf. Dieses sollte ebenfalls entfernt werden, bevor die Daten an einen Zustandserkenner weitergeleitet werden. Im Folgenden werden beispielhaft zwei Verfahren vorgestellt. Dies sind die Principal Component Analysis und selbstorganisierende Karten bzw. Kohonen Karten. Ihre Wirkungsweise wird erläutert und anschließend werden jeweils praktische Anwendungen der beiden Verfahren vorgestellt.

2 Multivariate Analyse

Als multivariate Analysemethoden (multivariate analysis) [4] bezeichnet man Verfahren, mit deren Hilfe sich multivariat verteilte statistische Variablen untersuchen lassen. Das bedeutet man betrachtet nicht die Wahrscheinlichkeitsverteilung einzelner Variablen, sondern die gemeinsame Verteilung mehrer Zufallsvariablen. So lassen sich Zusammenhänge und Abhängigkeiten zwischen verschiedenen Datenreihen bzw. Merkmalen finden.

Bekannte multivariate Analysemethoden sind z.B. die Clusteranalyse, Diskriminanzanalyse oder auch die Independent Component Analysis, eine schärfere Form der im Folgenden vorgestellten Principal Component Analysis [7].

Bei der Merkmalsextraktion und -selektion liegt das Hauptaugenmerk in der Regel auf der Minimierung von Zusammenhängen und Abhängigkeiten zwischen Daten. Für effiziente Zustandserkenner, deren Berechnungen häufig sehr aufwendig sind, ist es notwendig die Anzahl der Daten welche an den Erkenner gehen möglichst gering zu halten. Da Abhängigkeiten in Daten in der Regel auf redundante Information hinweisen, möchte man diese minimieren. Zudem lassen sich zwischen zusammenhangsfreien Merkmalen wesentlich einfacher Regel finden, welche es einem Zustandserkenner ermöglichen einzelne Benutzer- und Umgebungszustände voneinander abzugrenzen und somit eindeutig zu Erkennen.

2.1 Principal Component Analysis

Durch Sensoren gewonnene Daten lassen sich grundsätzlich als Vektoren darstellen. Hat man zum Beispiel in einer Stadt 150 Sensoren aufgestellt die pro Stunde die Verkehrsdichte an ihrem Standort messen, so erhält man pro Stunde einen 150-dimensionalen Vektor. Diesen Vektor nennt man auch Merkmalsvektor, da seine Einträge 150 Datenreihen bzw. Merkmale definieren. Für jeden der 150 Sensoren eines. Mittels der Principal Component Analysis (PCA) oder auch Hauptkomponentenanalyse werden jetzt neue Vektoren gesucht, mit deren Hilfe sich die ursprünglichen Vektoren als Linearkombination der neuen Vektoren darstellen lassen. Die neuen Vektoren werden auch als Hauptvektoren bezeichnet.

Durch Anwendung der Hauptvektoren auf die ursprünglichen Datenreihen bzw. Merkmale erhält man neue Datenreihen. Diese Datenreihen nennt man Hauptkomponenten und sie sind Linearkombinationen der ursprünglichen Datenreihen bzw. Merkmale. Zudem sind die Hauptkomponenten dekorreliert, also linear unabhängig und lassen sich nach ihrer Wertigkeit ordnen. Auf das Verkehrsdichtebeispiel bezogen existieren 150 Sensoren die Daten liefern. Aus diesen

Daten lassen sich genauso viele Hauptkomponenten wie ursprünglich vorliegende Datenreihen extrahieren, also 150. Ordnet man diese jetzt ihrer Wichtig entsprechend an, reichen in der Regel aber schon die ersten, wichtigsten Hauptkomponenten aus, um die ursprünglichen Datenreihen hinreichend zu rekonstruieren.

Für die Hauptvektoren der PCA gilt, dass der erste und somit wichtigste Vektor in Richtung der größten Varianz der Ursprungsvektoren zeigt. Für alle weiteren Hauptvektoren gilt, dass sie in Richtung der größten Varianz zeigen unter der Bedingung, dass sie orthogonal zu den vorherigen Hauptvektoren sind. Die Wichtigkeit der Hauptvektoren definiert sich direkt über die Varianzen der Ursprungsvektoren. Die Varianz beschreibt dabei die Streuung der Daten, also wie stark die Werte einer Datenreihe vom Mittelwert der Daten abweichen.

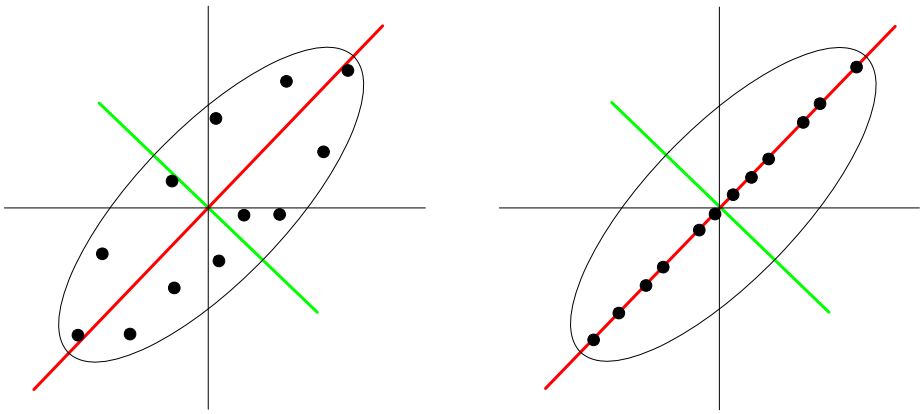


Abbildung 1. Geometrische Interpretation der Principal Component Analysis für zwei Merkmale.

Im Falle von zwei Merkmalen lassen sich die Ursprungsvektoren als Punkte in einem zwei-dimensionalen Koordinatensystem darstellen, wie es in Abb. 1 links zu sehen ist. Abgebildet ist eine ellipsenförmige Punktwolke. Der erste Hauptvektor ist in rot eingezeichnet und der zweite in grün. Abbildung 1 rechts zeigt die rekonstruierten Punkte, wenn nur der erste Hauptvektor zur Rekonstruktion verwendet wird. In der Tatsache das Merkmale, welche eine große Varianz aufweisen, wesentlich mehr zum Informationsgehalt erhobener Daten beitragen als solche mit geringer Varianz, liegt die wirkungsweise der PCA begründet. Die PCA extrahiert aus den ursprünglichen Merkmalen genau solche, die eine hohe Varianz besitzen und ordnet sie absteigend.

Im allgemeinen Fall liegt demnach eine Punktwolke in einem n-dimensionalen kartesischen Koordinatensystem vor, dessen Achsen durch die PCA so rotiert werden, dass die Varianzen entlang der Achsen maximal werden. Aus diesem Grund wird die PCA auch Hauptachsentransformation genannt.

Zur Bestimmung der Hauptvektoren betrachtet man die einzelnen Merkmale bzw. Datenreihen als Zufallsvariablen. Die n Merkmale lassen sich also auch als n Zufallsvariablen X_n schreiben. Der Mittelwert einer Datenreihe entspricht dabei dem Erwartungswert der zugehörigen Zufallsvariable und wird zunächst von jedem einzelnen Wert abgezogen. Danach haben alle Zufallsvariablen X_n den Erwartungswert 0. Aus den zentrierten Zufallsvariablen lässt sich dann die zugehörige Kovarianzmatrix berechnen. Dazu müssen allerdings die Verteilungsparameter der einzelnen Zufallsvariablen X_n bekannt sein, was in der Regel nicht der Fall ist. Daher wird die Kovarianzmatrix in der Praxis häufig anhand einer Stichprobe, der durch die X_n repräsentierten Daten, geschätzt.

Die Kovarianz ist eine Maßzahl, die Informationen über den linearen Zusammenhang zweier Zufallsvariablen X und Y angibt:

- Ein positiver Wert bedeutet, zwischen X und Y besteht ein proportionaler Zusammenhang.
- Ein negativer Wert bedeutet, zwischen X und Y besteht ein umgekehrt proportionaler Zusammenhang.
- Eine Kovarianz von 0 bedeutet, zwischen X und Y besteht kein linearer Zusammenhang.

Eine Kovarianzmatrix enthält alle Kovarianzkombinationen aus den X_n Zufallsvariablen, wobei auf der Hauptdiagonalen genau die Varianzen der X_n liegen. Die Hauptvektoren sind dann genau die Eigenvektoren der Kovarianzmatrix. Die Eigenwerte der Hauptvektoren entsprechen dabei gerade der Varianz, der aus den Hauptvektoren abgeleiteten Hauptkomponenten und definieren damit die Ordnung der Hauptkomponenten.

2.2 Anwendungen der Principal Component Analysis

2.2.1 Gesichtserkennung mittels Principal Component Analysis

Turk und Pentland haben 1991 in „Eigenfaces for Recognition“ [2] das Eigenfaces oder auch Eigengesichter Verfahren vorgestellt, welches Gesichtserkennung auf Basis der PCA ermöglicht.

Als erste wird ein Satz Bilder benötigt welcher als Trainingsmenge dient. Das Eigengesichter Verfahren arbeitet dabei auf Graustufenbildern, die als Vektoren interpretiert werden. Hat ein Bild $x * y$ Pixel, lassen diese sich in einem P-dimensionalen Vektor v unterbringen, der $x * y = P$ Grauwerte W enthält.

$$v = \begin{pmatrix} W_{11} \\ W_{12} \\ \cdots \\ W_{1x} \\ W_{21} \\ \cdots \\ W_{yx} \end{pmatrix}$$

Ein einfacher Zustandserkenner könnte bereits auf diesen Vektoren arbeiten und versuchen die Ähnlichkeit zweier Bilder über eine einfache Metrik, wie z.B. den Euklidischen Abstand, zu bestimmen. Ein solcher Zustandserkenner wäre aber aufgrund der hohen Dimension P der Vektoren recht langsam. Außerdem sind sich Bilder von Gesichtern häufig sehr ähnlich, was die Unterscheidung verschiedener Gesichter zusätzlich erschwert. Deswegen wird die PCA eingesetzt um neue Merkmale zu extrahieren, die eine bessere Unterscheidung von Gesichtern ermöglichen.

Ausgehend von einem Satz Bilder, gespeichert in den Vektoren $\Gamma_1, \Gamma_2, \ldots, \Gamma_N$ wird zuerst ein Durchschnittsgesicht Ψ berechnet:

$$\Psi = \frac{1}{N} \sum_{i=1}^{N} \Gamma_i \tag{1}$$

Danach wird zu jedem Γ ein Differenzgesicht Φ gebildet:

$$\Phi_i = \Gamma_i - \Psi \tag{2}$$

Die Differenzgesichter werden als Spaltenvektoren in eine Matrix A geschrieben.

$$A = [\Phi_1 \cdots \Phi_N] \tag{3}$$

Aus dieser Matrix lässt sich dann die Kovarianzmatrix C berechnen, deren Eigenvektoren ν schließlich die gesuchten Hauptkomponenten liefern, die dann Eigenfaces oder -gesichter genannt werden:

$$C = \frac{1}{N} \sum_{i=1}^{N} \Phi_i \Phi_i^T = AA^T \qquad C\nu = \lambda\nu \tag{4}$$

Abbildung 2 zeigt einige Eigengesichter. Es ist deutlich zu erkennen, dass sie wie schemenhafte Gesichter aussehen. Daher trägt das Verfahren auch den Namen Eigengesichter.

In der Praxis würde es allerdings viel zu lange dauern alle P Eigenvektoren ν_j zu berechnen, da dies für ein Bild der Auflösung 128x128 Pixel z.B. schon 16384 wären. Stattdessen errechnet man die Eigenvektoren μ_j einer neuen Matrix L:

$$L = A^T A \qquad L\mu = \lambda\mu \tag{5}$$

Die Dimension von L ist nur noch N und man kann zeigen das die Eigenwerte λ_j den N größten Eigenwerten von C entsprechen. Ferner gilt:

$$\nu_j = A\mu_j \tag{6}$$

Die N größten Eigenvektoren lassen sich also aus den Eigenvektoren μ der Matrix L berechnen.

Die Eigenvektoren ν_j werden dann entsprechend der Größe der Eigenwerte λ geordnet und noch normiert so dass gilt:

$$||\nu_j|| = 1 \tag{7}$$

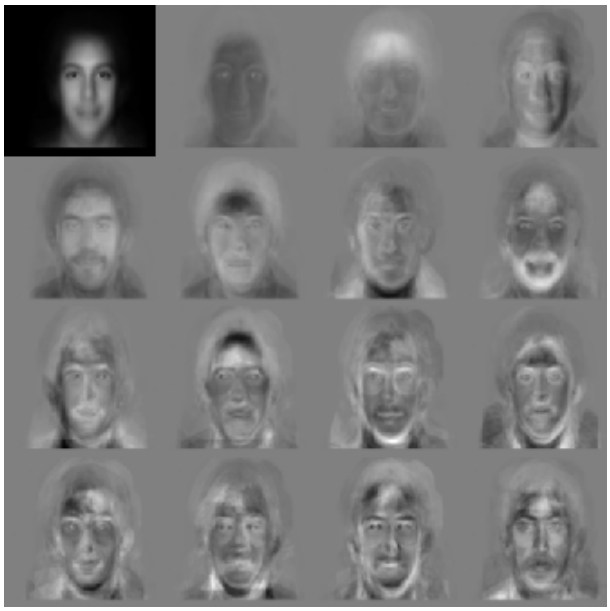


Abbildung 2. Eigengesichter aus einer Trainigsmenge von 128 Gesichtern.

Man wählt nun N' Eigenvektoren bzw. Hauptvektoren ν aus, deren Eigenwerte λ am größten sind. Diese Hauptvektoren spannen einen Unterraum auf, der auch Facespace oder Gesichtsraum genannt wird. In der Regel ist N' wesentlich kleiner als P und damit die Anzahl der zur Klassifizierung benötigten Hauptkomponenten ebenfalls.

Um ein Bild Γ nun zu Klassifizieren, muss es zunächst in den Gesichtsraum überführt werden. Dabei wird das Ursprungsbild in seine Eigengesichter Komponenten zerlegt:

$$\omega_j = \nu_j^T(\Gamma - \Psi) \qquad j = 1, \ldots, N' \tag{8}$$

Nun kann der Vektor $\Omega^T = [\omega_1 \cdots \omega_{N'}]$ mit Vektoren Ω_k von bereits bekannten Gesichtern verglichen werden. Dazu eignet sich zum Beispiel der Euklidische Abstand:

$$\varepsilon_k = ||\Omega - \Omega_k|| \tag{9}$$

Ein Bild Ω wurde als Gesicht k erkannt, wenn ε minimal ist. Überschreitet ε einen zuvor festgelegter Schwellenwert Θ, so kann davon ausgegangen werden, dass es sich bei Ω um kein Bild eines Gesichtes handelt.

Ein Nachteil der Eigengesichter ist die Tatsache, dass alle Bilder in der gleichen Auflösung vorliegen und ein frontal aufgenommenes Gesicht enthalten müssen. In der Praxis werden die wenigstens Bilder diese Bedingungen erfüllen. Es ist jedoch mit Hilfe des Gesichtsraumes ebenfalls möglich ein Gesicht in einem Bild zu finden. Dazu bildet man für gleichmäßig über einem Bild verteilte Punkte mit einer Umgebung Δ ein lokales Teilbild und bestimmt den Abstand

zum Gesichtsraum. Danach sucht man das Minimum von $\Delta(x, y)$ und hat die Stelle im Bild gefunden die ein Gesicht erhält.

Bei einer Bildauflösung von 128x128 Pixel erreicht man bereits eine sehr gute Erkennungsrate. Auf aktueller Hardware dauert dabei das Erkennen eines einzelnen Bildes nur wenige Millisekunden. In praktischen Versuchen am MIT konnte ein Rate von drei Personenidentifikationen pro Sekunde erreicht werden, inklusive dem Auffinden eines Gesichtes in einem größerem Bild.

Das Eigengesichter Verfahren ist allerdings anfällig gegen sich ändernde Lichtverhältnisse, variierende Kopfgrößen und die Ausrichtung eines Gesichtes im Bild. Bei wechselnden Lichtverhältnissen fällt die Erkennungsrate auf durchschnittlich 96%, was immernoch ein sehr gutes Ergebnis ist. Stärker wirkt sich schon die Ausrichtung des Gesichtes aus. Bei verschieden geneigten Gesichtern fällt die Erkennungsrate auf durchschnittlich 85%. Sie lässt sich aber durch Rotation des Bildes bei gleichzeitigem Minimieren des Abstands des Bildes zum Gesichtsraum wieder verbessern. Am empfindlichsten reagiert das Verfahren auf unterschiedliche Kopfgrößen. Hier fällt die Erkennungsrate auf durchschnittlich 64%. Es ist aber möglich, das Bild zu skalieren und für verschiedene Größen die Erkennung durchzuführen.

Auf diese Weise lassen sich grundsätzlich gute Ergebnisse mit dem Eigengesichterverfahren erzielen. Bei Seitlich von der Kamera abgewandten Gesichtern, bricht die Erkennungsrate jedoch rapide ein, ohne eine Möglichkeit diese wieder zu steigern. In dem Fall Hilft nur eine Lösung mit mehreren Kameras.

2.2.2 Automatische Ausrichtungskalibrierung mittels Principal Component Analysis

In „A New Method for Auto-Calibrated Object Tracking“ [1] wird ein System zur Objektverfolgung mittels Ultraschall beschrieben. Über sechs im Raum verteilte Empfänger können ein oder mehrere kleine Sender verfolgt werden. Ein wichtiger Aspekt des Systems ist, dass es in der Lage ist sich selbst zu kalibrieren. Damit soll erreicht werden, dass die Empfänger möglichst frei in einem Raum verteilt werden können und später kein manuelles und meistens aufwendiges Bestimmen der Abstände und relativen Positionen der Empfänger zueinander durch den Benutzer notwendig ist.

Die automatische Kalibrierung des Systems erfolgt, indem ein einzelner Sender für ca. 30 Sekunden im zu vermessenden Raum bewegt wird. Dabei werden kontinuierlich die Positionsdaten des Senders aufgezeichnet und aus den gewonnen Daten ein Koordinatensystem bestimmt, welches die Positionen der einzelnen Empfänger enthält. Der Ursprung des Koordinatensystems kann vor der Kalibrierung relativ zu einem beliebigem Empfänger durch den Benutzer festgelegt werden. Die Ausrichtung der Achsen nach der automatischen Kalibrierung ist allerdings willkürlich vom System festgelegt. Da der Benutzer des Systems das verwendete Koordinatensystem ohne größere Einschränkungen selbständig bestimmen können soll, ist das System tatsächlich bewusst dafür ausgelegt mit beliebig ausgerichteten Koordinatensystemen zu arbeiten solange diese orthogonal sind.

An dieser Stelle kommt die PCA zum Einsatz, um es dem Benutzer zu ermöglichen auf einfache Weise ein eigenes Koordinatensystem festzulegen, dessen Achsen z.B. parallel zu den Wänden eines Raumes ausgerichtet sind. Dazu muss der Benutzer mit einem Sender in der Hand einen L-förmigen Pfad im Raum ablaufen, wie er in Abb. 3 dargestellt ist. Dieser Pfad bestimmt die neue x- und y-Achse des Systems.

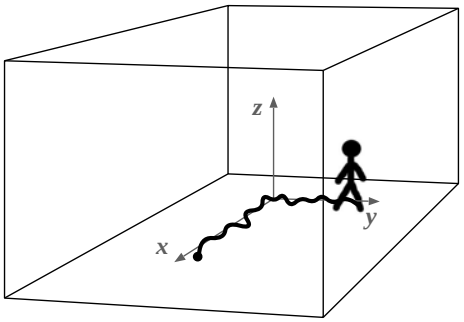


Abbildung 3. Der Benutzer bewegt sich auf einem L-förmigen Pfad, welcher die x- und y-Achse des Systems bestimmt.

Abbildung 4 (links) zeigt die durch den Benutzer auf dem Pfad gesammelten Daten im durchs System bestimmten Koordinatensystem. Nun wird eine PCA auf den Vektoren $X = (X_1, X_2, \ldots, X_n)$ ausgeführt. Dazu wird zunächst der Mittelwert $\bar{X}$ berechnet und von den Punkten abgezogen. Danach wird die zu X gehörige Kovarianzmatrix M bestimmt. Da es sich um drei-dimensionale Punkte und somit um drei Datenreihen handelt, liefert die Kovarianzmatrix drei Eigenvektoren e_1, e_2 und e_3. Nach den Eigenwerten geordnet, zeigen dann die beiden größten Eigenvektoren in Richtung der vom Benutzer abgelaufenen x- und y-Achse. Der dritte Eigenvektor zeigt in Richtung der z-Achse. Abbildung 4 (mitte) zeigt die Projektion der Daten mit Hilfe der normierten Eigenvektoren $\hat{e}$ in die XY-Ebene durch:

$$X_i' = [\hat{e}_1, \hat{e}_2, \hat{e}_3]^T \, (X_i - \bar{X}) \tag{10}$$

Schwankungen der Punkte in z-Richtung können an dieser Stelle bereits durch Ignorieren des dritten Eigenvektors bei der Projektion in die XY-Ebene eliminiert werden.

Um das System robuster zu machen, werden die Schwankungen in z-Richtung aber erst in einem zweiten Schritt mit einem anderen Verfahren entfernt, welches auch die Rotation der Daten um die z-Achse und deren Verschiebung vom Ursprung aus bestimmt. Abbildung 4 (rechts) zeigt das Endgültige Ergebnis, bei dem der L-förmige Pfad an den Achsen des ursprünglichen Koordinatensystems ausgerichtet ist und im Ursprung liegt.

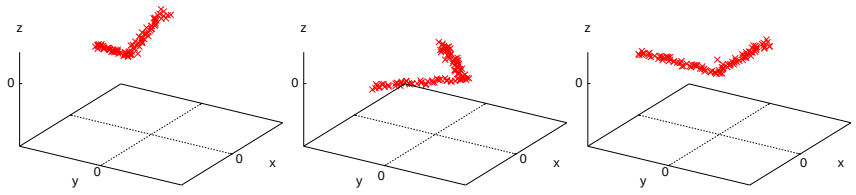


Abbildung 4. L-förmiger vom Benutzer bestimmter Pfad: (links) Im internen Koordinatensystem. (mitte) Projiziert in die XY-Ebene. (rechts) An den Achsen ausgerichtet.

Damit ist eine Transformation gefunden, um von Sendern empfangene Positionen vom System- ins Benutzerkoordinatensystem zu überführen und umgekehrt.

Die Fehlkalibrierungsrate des Systems liegt praktisch bei null. Solange der Benutzer einen hinreichend großen L-förmigen Pfad abläuft, sind die Schwankungen der Daten in z-Richtung grundsätzlich wesentlich geringer, als auf der x- und y-Achse. Ebenfalls kann in diesem Fall davon ausgegangen werden, dass die Varianz der Daten in Laufrichtung immer deutlich größer ist als orthogonal zur Laufrichtung. Welcher Teil des "L"s durch die PCA als x- und welcher als y-Achse erkannt wird, ist ebenfalls irrelevant. Dies kann auch später noch festgestellt werden, wenn man z.B. annimmt, dass der L-förmige Pfad auf der x-Achse beginnt und auf der y-Achse endet. Benötigt werden nur die ersten zwei Hauptvektoren, aus denen sich der Dritte bereits automatisch ergibt.

3 Selbstorganisierende Karten

Selbstorganisierende Karten oder Kohonen Self-Organizing Maps (KSOM) sind eine Entwicklung des finnischen Professors Teuvo Kohonen [9]. Sie werden häufig auch Kohonen Maps oder Kohonennetze genannt.

Die grundlegende Eigenschaft der KSOM ist es, hochdimensionale Eingabedaten und nicht-lineare statistische Abhängigkeiten zwischen diesen Daten mit Hilfe einfacher geometrischer Beziehungen darzustellen. In der Regel handelt es sich dabei um zwei-dimensionale Karten auf denen Eingabedaten mit ähnlichen Merkmalen Positionen zugeordnet werden, die dicht nebeneinander liegen. Als Ergebnis liefert das Verfahren demnach stark komprimierte Ausgabedaten, deren Dimension gegenüber den Eingabedaten wesentlich reduziert ist. Gleichzeitig bleibt aber die Topologie der Eingabedaten und metrische Beziehungen zwischen den Eingabedaten weitgehend erhalten. Eine Einteilung der Ausgabedaten in Gruppen lässt sich anschließend z.B. durch verschiedene auf die Karte angewendete Clustering-Algorithmen realisieren.

Die Idee der Kohonenkarten ist dabei der Biologie entliehen. Durch Untersuchungen der Arbeitsweise des menschlichen Gehirns wurde festgestellt, dass die Großhirnrinde an vielen Stellen eine Struktur aufweist, bei der unterschiedliche

Eingaben wie akustische, visuelle oder haptische Signale topologieerhaltend auf ähnliche Bereiche der Großhirnrinde abgebildet werden.

Die Umsetzung der KSOM erfolgt über künstliche neuronale Netze. Diese Netze bestehen in den meisten Fällen aus zwei Schichten. In Abb. 5 ist beispielhaft eine KSOM dargestellt. Die erste Schicht ist die Eingabeschicht, in Abb.5 (links) zu sehen. Für n-dimensionale Daten besteht die Eingabeschicht aus n Eingabeneuronen $p_1, p_2, \ldots, p_n$, die jeweils mit jedem Neuron der zweiten Schicht verbunden sind. Verbindungen der Eingabeneuronen untereinander existieren nicht.

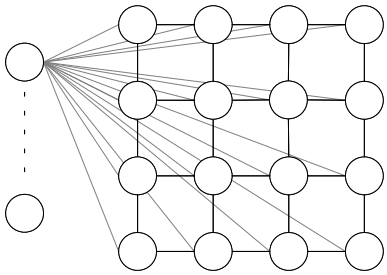


Abbildung 5. Eine zwei-schichtige KSOM bestehend aus Eingabe- und Ausgabeschicht. Sämtliche Eingabeneuronen sind mit jedem Ausgabeneuron verbunden.

Die zweite Schicht ist die Ausgabeschicht und die eigentliche Kohonen Karte. In Abb. 5 (rechts) besteht sie aus 16 untereinander verbundenen Neuronen q_{ij}. Je nach Anwendung der KSOM kann die Ausgabeschicht verschiedenste Formen annehmen. Satt vier Verbindungen pro Neuron, kann z.B. eine hexagonale Anordnung der Ausgabeneuronen gewählt werden mit sechs Verbindungen pro Neuron, ausgenommen am Rand der Karte liegende Neuronen. Oftmals verbindet man auch die Neuronen an den gegenüberliegenden Rändern der Karte miteinander, was im Prinzip eine torusförmige Ausgabeschicht ergibt.

Die Neuronen der Ausgabeschicht sind über hemmende Verbindungen miteinander verbunden. Dies entspricht dem Prinzip der lateralen Hemmung bzw. Umfeldhemmung aus der Biologie. Einfach ausgedrückt unterdrückt ein aktives Neuron die Aktivität in anderen Neuronen, je weiter diese vom aktiven Neuron entfernt sind, während die Aktivität von Neuronen in direkter Nähe des aktiven Neurons verstärkt wird.

Des Weiteren hat jedes Neuron q_{ij} der Ausgabeschicht einen n-dimensionalen Gewichtsvektor w_{ij}, dessen Dimension der Dimension der Eingabedaten entspricht. Außerdem noch eine festgelegte Position innerhalb der Ausgabeschicht. Die Positionen der Ausgabeneuronen und ihre Verbindungen definieren somit implizit die Kartenstruktur. Für eine hexagonale oder torusförmige Ausgabeschicht ist unter Umständen eine Anpassung der Indizes der Ausgabeneuronen und Gewichtsvektoren nötig. Wichtig ist jedoch nur das diese eindeutig identifi-

ziert werden können. Im Folgenden wird von einem quadratischen Gitter ausgegangen, in dem die Neuronen der Ausgabeschicht angeordnet sind.

Bevor Daten Klassifiziert werden können, muss die KSOM trainiert werden. Zunächst werden dazu die Gewichte w_{ij} der Neuronen q_{ij} initialisiert. Dies kann mit zufälligen, kleinen Werten, aber auch nach einem bestimmtem dem jeweiligen Problem angepasstem Verfahren geschehen. Anschließend wird eine Trainingsmenge T von n-dimensionalen Trainingsvektoren t_k gewählt. Entsprechend einer Wahrscheinlichkeitsfunktion werden nun einzelne Vektoren aus T nacheinander ausgewählt. Dies kann erneut völlig zufällig oder z.B. im Eingaberaum gleichverteilt geschehen.

Der gewählte Trainingsvektor t wird dann über eine definierte Metrik, im allgemeinen der Euklidische Abstand, mit allen Gewichten w_{ij} verglichen und das Neuron mit dem geringsten Abstand als Best Matching Unit (BMU) ausgewählt:

$$q_{\text{BMU}} = \min_{ij} ||t - w_{ij}|| \tag{11}$$

Der Gewichtsvektor der BMU q_{BMU} und die Gewichtsvektoren der Nachbarsneuronen von q_{BMU} werden dann, gemäß einer definierten Lernrate α, in Richtung von t angepasst. Die Gewichte der Nachbarn werden dabei weniger stark angepasst, als das Gewicht von q_{BMU} selbst. Die Nachbarschaft definiert man über eine Funktion θ. Dieser Vorgang wird so lange wiederholt bis jeder Trainingsvektor t_k einmal an die KSOM übergeben wurde. Man nennt diesen Vorgang auch unüberwachtes Lernen, da keinerlei Zielwerte oder Abbruchkriterien existieren, welche den Lernprozess steuern.

Die Lernrate α legt fest, wie stark ein Gewichtsvektor w_{ij} während eines Traningsschrittes verändert wird und somit wie schnell ein Neuron q_{ij} lernt:

$$\Delta w_{ij} = \alpha ||t - w_{ij}|| \tag{12}$$

Meistens ist die Lernrate α dabei Abhängig von der Zeit. Dadurch lernen die Neuronen am Anfang sehr schnell, so dass sich eine grobe Struktur der Karte herausbilden kann. Gegen Ende des Lernprozesses erfolgt dann nur noch eine feinere Anpassung der Karte. Eine lineare Lernrate wäre dabei folgendermaßen definiert:

$$\alpha(\tau) = \alpha_0 \left(1 - \frac{\tau}{\tau_{\max}}\right) \tag{13}$$

Mit α_0 als Startwert der Lernrate und $\tau_{\max}$ als Endzeitpunkt der Trainingsphase. Je nach Anwendung werden z.B. auch logarithmische oder exponentielle Lernraten verwendet.

Die Nachbarschaft θ kann, ähnlich wie die Lernrate α, über verschiedene Funktionen definiert werden. Im Allgemeinen wird die Gaussglocke verwendet und θ folgendermaßen definiert:

$$\theta(q_{ij}, q_{\text{BMU}}) = e^{-\left(\frac{\Delta(q_{ij}, q_{\text{BMU}})}{N}\right)^2} \tag{14}$$

N gibt hier den Nachbarschaftradius an und wird, analog zur Lernrate, abhängig von der Zeit gemacht:

$$N(t) = N_0 \, e^{-\frac{\tau}{\tau_{\max}}} \tag{15}$$

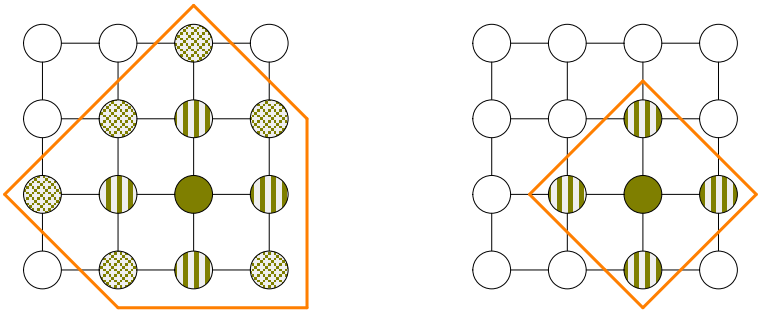


Abbildung 6. Best Matching Unit mit Nachbarschaftsradius $N = 2$ (links) und $N = 1$ (rechts).

Abbildung 6 zeigt beispielhaft eine gefundene Best Matching Unit und ihre beeinflussten Nachbarn im Laufe des Lernprozesses.

Insgesamt ergibt sich demnach folgende Formel zur Bestimmung der Gewichtsvektoren zum Zeitpunkt τ nachdem der Trainingsvektor t_k an die KSOM übergeben wurde:

$$w_{ij}(\tau) = w_{ij}(\tau - 1) + \alpha(\tau)\theta(q_{ij}, q_{\text{BMU}})||t_k - w_{ij}(\tau - 1)|| \tag{16}$$

Die fertig Trainierte KSOM kann nun z.B. verwendet werden, um über den Abstand verschiedener Eingabevektoren auf der Karte Rückschlüsse über deren Ähnlichkeit zu ziehen. Häufig Verwendet man KSOM auch um Zusammenhänge zwischen hochdimensionalen Daten für den Menschen anschaulich Darzustellen, indem man eine zwei-dimensionale Karte um eine dritte Dimension erweitert. Dazu berechnet man den durchschnittlichen Abstand u_{ij} des Gewichtes w_{ij} eines Neurons q_{ij} zu den Gewichten der benachbarten Neuronen der Karte. Anschließend färbt man die Positionen der Neuronen q_{ij} in Karte entsprechend der zugehörigen u_{ij} ein, nachdem man die u_{ij} normiert und auf eine beliebige Farbskala übertragen hat. In Abb. 7 sind einige Beispiele für solche Visualisierungen zu sehen, die mit dem Programm Synapse [5] erstellt wurden.

3.1 Anwendungen Selbstorganisierender Karten

3.1.1 Adaptive Kontextsensitivität durch Selbstorganisierende Karten

„What Shall We Teach Our Pants?“ [10] beschäftigt sich mit der Frage, wie adaptive Kontextsensitivität (context awareness) erreicht werden kann. Damit ist eine auf den einzelnen Benutzer zugeschnittene Kontextsensitivität gemeint, die sich an das Verhalten des Benutzers anpasst. Als Beispiel für existierende

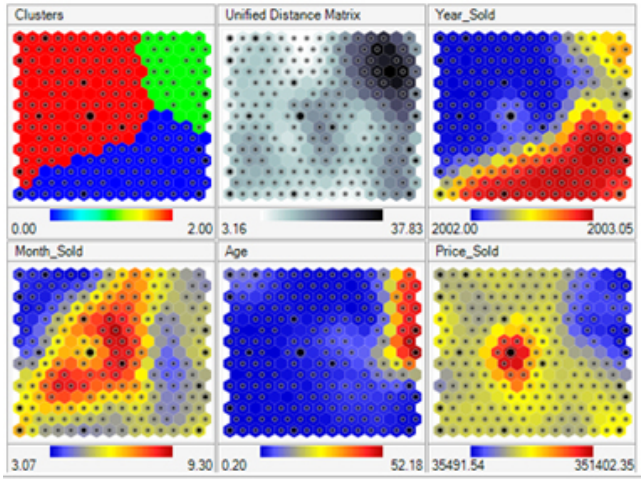


Abbildung 7. Visualisierungen verschiedener Kohonen Karten mit dem Programm Synapse.

kontextsensitive Systeme werden mobile Geräte angeführt, die ihre Bildschirmhelligkeit abhängig von den Lichtverhältnissen regeln. Für ein solches Gerät kann vorab der Zusammenhang zwischen der Ausgabe des Lichtsensors und der einzustellenden Bildschirmhelligkeit festgelegt werden. Viele Benutzer- und Umgebungszustände variieren jedoch abhängig vom Benutzer und der Umgebung in der sie eingesetzt werden sehr stark. Das macht Systeme nötig, in denen der Benutzer und nicht der Designer des Systems festlegen kann, welche Umgebungs- und Benutzerzustände überhaupt existieren und erkannt werden sollen.

Abbildung 8 zeigt den Aufbau des vorgestellten Systems zur adaptiven Kontextfindung. Im ersten Schritt werden mittels verschiedener Sensoren möglichst umfassende Informationen über die Umgebung und den Benutzer gesammelt.

Im nächsten Schritt findet eine Vorverarbeitung der erfassten Daten statt. Nicht alle der durch die Sensoren erfassten Daten werden direkt an die dritte Schicht weitergeleitet. Bei einigen Sensoren kann es vorteilhaft sein, einfache Berechnungen wie z.B. Standard Abweichung, Streuung oder Fast Fourier Transformation auf ihren Daten durchzuführen. Dadurch können mitunter aus den Daten eines einzelnen Sensors verschiedene Merkmale gewonnen werden. Von einem Lichtsensor aufgenommene Daten lassen sich beispielsweise an dieser Stelle bereits in zwei Datenreihen trennen, welche die Lichtintensität und -frequenz wiederspiegeln.

Im dritten Schritt werden dann schließlich sämtliche Merkmale durch eine oder mehrere KSOM verarbeitet. Die Verwendung von KSOM in diesem Schritt liegt darin begründet, das angestrebte Ziel einer adaptiven Kontextsensitivität statt einer festvorgegebenen zu erreichen. Dadurch das eine KSOM unüberwacht

lernt, ist das System in der Lage sich an die Umgebung in der es eingesetzt wird anzupassen, ohne das ein Eingreifen des Benutzers erforderlich ist. Es wird eine drei-dimensionale Aktivitätslandkarte erstellt, indem über den x und y Werten eines Neurons der Kohonen Karte abgetragen wird, wie oft das entsprechende Neuron als BMU ausgewählt wurde. Eine solche Aktivitätslandkarte ist in Abb. 9 dargestellt.

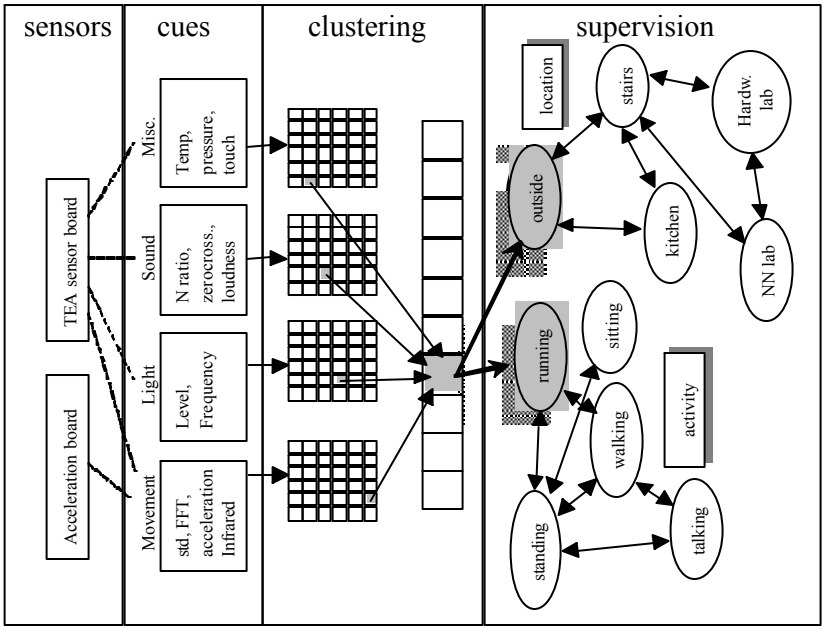


Abbildung 8. Aufbau des Systems: Erfassung von Sensordaten, Vorverarbeitung der Sensordaten, Verarbeitung durch KSOM, Auswertung der KSOM und Ausgabe des Umgebungs- und Benutzerzustands.

Anhand dieser Karte, welche sich einfach visualisieren lässt, kann der Benutzer des Systems nun selber Marken setzten, welche verschiedene für ihn interessante Kontexte kennzeichnen. Dies ist im letzten Schritt in Abb. 8 zu sehen. Im Folgenden entscheidet das System dann selbständig welche Marken am ehesten der Ausgabe der KSOM entsprechen.

Ein Nachteil der KSOM ist allerdings, dass diese Anfangs zwar sehr schnell lernen, nach einer gewissen Zeit aber relativ starr bleiben und kaum noch dazulernen. Das kann zu einem Problem werden, wenn ein System nicht einmal auf eine Umgebung und einen Benutzer trainiert werden soll, sondern dauerhaft adaptiv bleiben muss. Ein weitere Nachteil ist, das die KSOM mit einer steigenden Anzahl an Eingabemerkmalen, wie die meisten Verfahren, immer langsamer

lernt, was mitunter ein langwieriges Training des Systems erfordern kann, bevor es zuverlässige Ergebnisse liefert.

Es wird versucht diese Probleme zu umgehen, indem eine Hierarchie mehrerer KSOM eingesetzt wird. Zum einen um die Menge an Merkmalen die von einer einzelnen KSOM verarbeitet werden müssen zu reduzieren. Zum anderen damit es möglich ist einzelne Karten teilweise zurückzusetzen und somit die Adaptivität des Systems zu erhalten, ohne sämtliche bereits gelernten Kontexte zu vergessen. Dieses Vorgehen birgt aber neue Probleme, wie z.B. die Frage welche KSOM zu welchem Zeitpunkt repräsentativ für die Kontexterkennung ist. Für diese und andere Probleme ist teilweise noch keine endgültige Lösung gefunden worden.

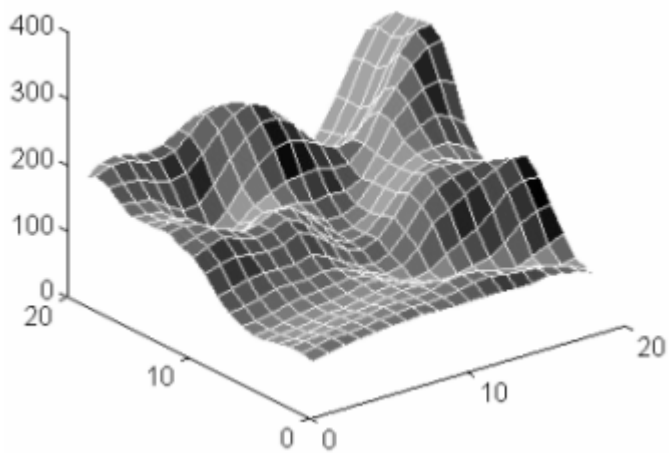


Abbildung 9. Eine Aktivitätslandkarte.

Als Prototyp des Systems wurde eine Hose mit Beschleunigungs- und Positionssensoren ausgestattet. Bei diesem System muss der Benutzer während der Trainingsphase beim Sitzen, Laufen, Stehen oder z.B. Treppensteigen einen entsprechend beschrifteten Knopf drücken. Nach einer kurzen Trainingsphase werden diese Aktivitäten dann bereits relativ zuverlässig wiedererkannt. Für den Prototypen wurden nur zwei kombinierte Beschleunigungs- und Positionssensoren verwendet, die bereits ausreichen um trotz der ungelösten Probleme, gute Resultate zu liefern.

3.1.2 Gestenerkennung mit Selbstorganisierenden Karten

In „Gesture Classification with Hierarchically Structured Recurrent Self Organizing Maps“ [3] werden KSOM eingesetzt, um Bewegungsdaten zu verarbeiten und Gesten zu erkennen. Der Benutzer des Systems nimmt dafür einen kleinen Holzwürfel, den Gesture Cube, in die Hand und vollführt mit ihm vordefinierte Gesten, über die sich dann beliebige Anwendungen steuern lassen. Gedacht ist

der Würfel dabei als neuartiges Eingabegerät für Multimedia Geräte im Heimbereich. Genauso denkbar ist jedoch auch eine Anwendung in Mobiltelefonen oder eine Kombination mit einer klassischen Fernsteuerung.

Gestenerkennung ist ein recht ungewöhnlicher Einsatzbereich für KSOM, da diese in der Regel zur Visualisierung und Dimensionsreduktion von weiterzuverarbeitenden Daten eingesetzt werden. KSOM liefern jedoch auch für stark verrauschte Daten noch gute und robuste Ergebnisse, wodurch eine fehlende Vorverarbeitung von Sensordaten ausgeglichen werden kann.

Abbildung 10 zeigt das Innenleben des Gesture Cubes, welcher drahtlos mit einem PC verbunden wird und neben einem RF-Sender noch zwei zwei-achsige Beschleunigungssensor enthält. Im Gegensatz zu anderen ähnlichen Projekten findet bei diesem System keine Vorverarbeitung der Sensordaten im Würfel statt. Die Beschleunigungssensordaten werden also direkt an die KSOM übergeben.

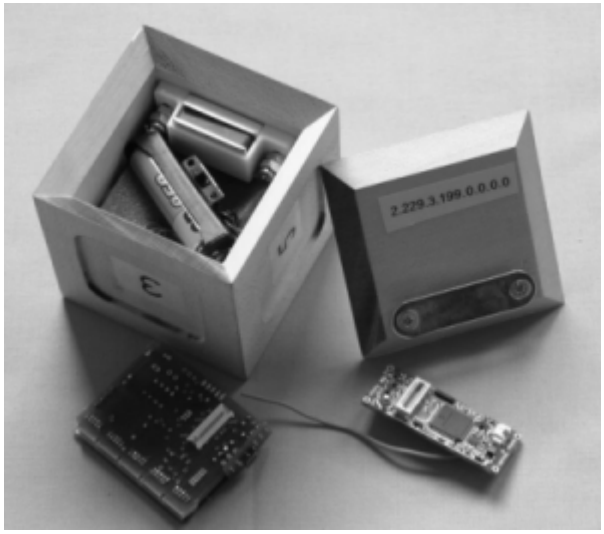


Abbildung 10. Der Gesture Cube enthält ein drahtloses Sensor System welches über eine RF-to-UDP Verbindung mit einem PC Verbunden werden kann.

Da der Gesture Cube während einer Bewegung kontinuierlich Daten sendet, sollen nicht einzelne Eingabevektoren durch die KSOM verarbeitet werden, sondern Sequenzen von zeitlich zusammenhängende Vektoren. Dazu wird das Konzept der KSOM dahingehend erweitert, dass eine Rückkopplung der Gewichtsvektoren aller angeregten Neuronen, also der BMU und ihrer Nachbarn, stattfindet. Der Zustand der Neuronen im nächsten Zeitschritt ergibt sich dann nicht nur aus dem aktuellen Eingabevektor, sondern ist ebenfalls abhängig von den abgeschwächten Gewichtsvektoren der BMU und ihrer Nachbarn im vorherigen Zeitschritt.

Für den Gesture Cube wurde die KSOM so angelegt, dass man als Ergebnis einer Eingabesequenz eine Karte mit einem Muster der zu den Eingabevektoren korrespondierenden Neuronen erhält, wobei der Grad ihrer Aktivierung der zeitlichen Reihenfolge ihrer Aktivierung während der Sequenz entspricht.

Der Gestenerkenner selbst ist aus mehreren Schichten dieser STORM (Strict Temporally Ordered Recurrent Map) genannten KSOMS zusammengesetzt. Jede Schicht verarbeitet eine festgelegte Anzahl an Eingabevektoren, bevor das daraus resultierende Aktivierungsmuster an die nächst höhere Schicht als Eingabe weitergeleitet wird. Dadurch ergibt sich in den höheren Schichten ein immer gröberes Bild, welches die Eingaben eines wachsenden Zeitraums repräsentiert. In der obersten Schicht werden dann die einzelnen Aktivierungsmuster aufsummiert, was schließlich eine Aktivierungsdichtekarte liefert. Die oberste KSOM stellt demnach die Häufigkeit verschiedener Impulse für einen festgelegten Zeitraum dar. Diese Daten werden an einen k-Nearest Neighborhood Klassifikator weitergleitet, der sie einer Klasse zuordnet. In diesem Fall also einer dem Gesture Cube zuvor antrainierten Geste.

In einem ersten Test wurde der Würfel mit 10 Gesten trainiert. Dabei wurde jede Geste ca. 10-20 mal wiederholt, wobei die Größe und Geschwindigkeit der Eingabe variiert wurde. Anschließend wurde die Erkennungsrate für verschiedene, oberhalb der KSOM sitzende Klassifizierungsalgorithmen getestet. Ein 5-nearest-neighbor Klassifikator erreichte nach der kurzen Trainingsphase die beste Erkennungsrate von bereits 80%. Mittels Support Vector Machine (SVN) wurden 76% erreicht und Radial Basis Function Networks (RBF Network) erreichten immerhin noch 73%. Nach ca. 90 Wiederholungen pro Geste erreicht der Gesture Cube mit Hilfe des k-nearest-neighbors Klassifikators eine Erkennungsrate von praktisch 100%.

3.1.3 Selbstorganisierende Karten zur Abhängigkeitsbestimmung zwischen Sensoren in Sensornetzwerken

In „Real-Time Analysis Of Correlations Between On-Body Sensor Nodes (with Topological Map)“ [8] wird analysiert, inwiefern es bei der Auswertung erfasster Daten hilfreich sein kann, die Lage und Zusammenhänge zwischen den Positionen der erfassenden Sensoren zu berücksichtigen. Dazu wird eine Vielzahl an Sensoren am menschlichem Körper, in diesem Fall an den Hosenbeinen einer Hose, befestigt, wie es in Abb. 11 zu sehen ist. Es handelt sich um 20 zwei-achsige Beschleunigungssensoren $S_1, \ldots, S_{20}$.

In großen Sensornetzwerken ist es häufig wichtig zu wissen, welche Daten von welchem Sensor aus dem Netzwerk stammen bzw. wo sich dieser gerade im Netzwerk befindet. Dazu ist es Notwendig z.B. die Sensoren selbst mit Positionserkennern auszustatten oder die Sensoren so zu platzieren, dass sich ihre relative Lage zueinander möglichst nicht ändert. Zudem kann man eindeutige IDs für jeden Sensor vergeben.

In der Praxis existieren aber Fälle, in denen z.B. aufgrund einer Vielzahl sich selbst vernetzender und organisierender Sensoren keine eindeutigen IDs vergeben werden können und die initiale Lage der einzelnen Sensoren nicht bekannt ist.

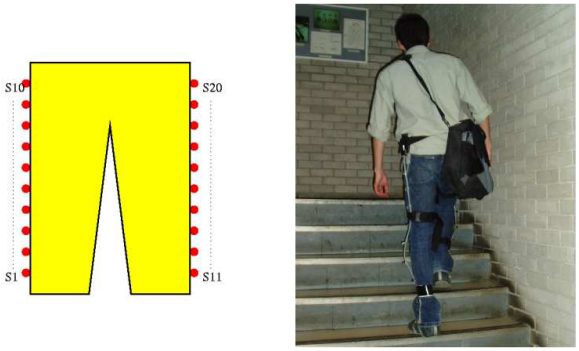


Abbildung 11. Verteilung der Beschleunigungssensoren an der Hose.

Dann benötigt man Verfahren, welche die Sensorpositionen im Netzwerk anhand der durch die Sensoren erfassten Daten bestimmen können.

Bei einer steigenden Anzahl an Sensoren pro Fläche, also einer steigenden Sensordichte, kann angenommen werden, dass dicht beieinanderliegende Sensoren ähnliche Daten liefern. Dann besteht zwischen den Daten der Sensoren ein linearer Zusammenhang, sie sind also miteinander korreliert.

Um diese Zusammenhänge zu untersuchen bildet man zunächst den Korrelationkoeffizienten der einzelnen S_i untereinander. Dieser gibt in einem Wertebereich von $[-1.0, \ldots, 1.0]$ lineare Zusammenhänge zwischen den Daten an. Dabei bedeutet 1.0 es existiert ein linearer Zusammenhang und 0.0 es existiert keiner. Ein Wert von -1.0 bedeutet einen umgekehrt proportionalen Zusammenhang zwischen den Daten.

Man errechnet Korrelationvektoren, indem jeweils ein Referenzsensor mit allen anderen verglichen wird und man die Ergebnisse in einen Vektor schreibt. Man erhält also 20 Korrelationsvektoren, für jeden Sensor einen, welche 40-dimensional sind. Denn die 20 Sensoren liefern insgesamt 40 Datenreihen bzw. Merkmale, also eines für jede Achse der zwei-achsigen Sensoren. Diese Vektoren werden als Eingabe für eine KSOM verwendet.

Anschließend wird auf der trainierten KSOM ein Growing Neural Gas (GNG) Algorithmus ausgeführt. Für eine Folge von Eingabedaten, welche durch die KSOM auf eine Folge von Positionen innerhalb der Karte abgebildet werden, erzeugt das GNG ein Netz von 20 Punkten, welche die einzelnen Sensoren repräsentieren. Verbindungen zwischen Punkten bedeuten, dass zwischen diesen Sensoren ein linearer Zusammenhang besteht.

Mit Hilfe der Independent Component Analysis (ICA) lassen sich aus den Korrelationsvektoren ebenfalls die Abhängigkeiten zwischen den einzelnen Sensoren bestimmen. Die ICA ist allerdings zu Rechenaufwendig um in Echtzeitanwendungen zum Einsatz zu kommen. Vergleicht man die durch ICA und GNG erhaltenen Daten, so fällt auf, dass diese die gleiche Struktur aufweisen, wobei die über KSOM und GNG erhaltenen Daten wesentlich gröber sind. Dies zeigt

jedoch, dass sich tatsächlich die Struktur eines Sensornetzwerk aus einer KSOM und den erfassten Daten ableiten lässt, ohne das weitere Strukturinformationen des Sensornetzwerks benötigt werden.

Im Gegensatz zur ICA eignet sich die KSOM sehr gut für einen Einsatz in Echtzeitanwendungen und erlaubt es die Struktur des Sensornetzwerks für einen Benutzer anschaulich zu visualisieren. So kann ein Benutzer z.B. defekte Knoten oder Änderungen an der Struktur eines Sensornetzwerkes visuell dargestellt bekommen.

4 Zusammenfassung

Für eine möglichst umfassende Kontextsensitivität ist es notwendig, mit Hilfe vieler verschiedener Sensoren Umgebungs- und Benutzerdaten ubiquitärer Systeme zu erfassen. Anhand der gesammelten Daten sollen Zustandserkenner Rückschlüsse über den Umgebungs- und Benutzerzustand ziehen, in dem ein ubiquitäres Gerät betrieben wird, um anschließend mit einer passenden Aktion zu reagieren. Durch neue und verbesserte Sensoren wachsen die Möglichkeiten die Umwelt eines ubiquitären Systems zu erfassen theoretisch stetig an. In der Praxis jedoch entsteht dadurch eine riesige Menge an Information, die kein Zustandserkenner schnell genug verarbeiten kann, um in angemessener Zeit auf Umwelteinflüsse und Anforderungen eines Benutzers zu reagieren.

Trotz ebenfalls stetig wachsender Rechenleistung benötigt man deshalb Verfahren, die in der Lage sind hochdimensionale, viele Merkmale enthaltende Daten auf einige wenige wichtige Merkmale zu reduzieren. Das ist die Aufgabe der Merkmalsextraktion und -selektion. Sie stellen einen zwingend notwendigen Schritt auf dem Weg von erfassten Sensordaten zur Kontextsensitivität dar.

Merkmalsextraktion und -selektion erfüllen dabei mehrer wichtige Aufgaben. Einmal werden unwichtige von wichtigen Merkmalen getrennt und im Idealfall nur relevante Daten an einen entsprechenden Zustandserkenner weitergeleitet. Oft weisen durch Sensoren erfasste Daten aber auch ein hohes Rauschen auf, welches während der Merkmalsextraktion herausgefiltert werden, um die Leistungsfähigkeit eines Zustandserkenners stark zu verbessern.

In vielen Fällen sind erfasste Merkmale auch ungeeignet zu Weiterverarbeitung und es können neue Merkmale aus ihnen gewonnen werden, welche sich besser zur Zustandserkennung eignen. Um die durch einen Zustandserkenner zu verarbeitende Datenmenge weiter zu reduzieren, versucht man ebenfalls bestehende Abhängigkeiten zwischen den erfassten Daten zu minimieren, da diese letztendlich nur redundante Information darstellen.

Zur Merkmalsextraktion und -selektion stehen eine Vielzahl an Verfahren zur Verfügung, von denen nur zwei exemplarisch behandelt worden sind. Allen gemein ist jedoch, dass man möglichst Ressourcen schonende Verfahren benötigt, welche innerhalb kurzer eine starke Reduktion aller durch Sensoren erfassten Daten eines Systems erreichen.

Welches Verfahren zum Einsatz kommt muss dabei je nach Problemstellung neu entschieden werden. Die hier vorgestellte PCA eignet sich sehr gut um aus

hochdimensionalen Daten neue Merkmale zu extrahieren und nur die wichtigsten der neuen Merkmale zu selektieren. Dabei werden die Daten von linearen Abhängikeiten befreit. Nicht-lineare Zusammenhänge bleiben jedoch bestehen. Dafür lässt sich die PCA wesentlich schneller berechnen als z.B. die Independent Component Analysis, welche Daten auch von nicht-linearen Zusammenhängen befreit. Wie das Beispiel der Eigengesichter zeigt, reicht jedoch die PCA bereits, um überzeugende Ergebnissse zu erhalten. Ebenfalls wird aber am Beispiel der Eigengesichter deutlich, das die Leistung der PCA mit stark verrauschten Daten schnell abnimmt.

Die KSOM liefern im Gegensatz dazu auch bei verrauschten Daten noch gute Ergebnisse, wie das Beispiel des Gesture Cube zeigt. Dafür wird hier zunächst eine Trainingsphase benötigt, deren Komplexität und Langwierigkeit ebenfalls mit der dimensionalität der zu Verarbeitenden Daten ansteigt. Dafür sind KSOM sehr gut geeignet um beliebig-dimensionale Daten topologieerhaltend auf zwei- oder drei-dimensionale Daten abzubilden und das auch noch sehr schnell, sobald eine KSOM erstmal trainiert ist. Das macht sie überall dort zum Idealen Verfahren, wo Daten für den Menschen anschaulich aufbereitet und dargestellt werden sollen. Zudem ermöglichen sie einem ubiquitären System dem Benutzer adaptive Kontextsensitivität zu bieten. Damit sind Benutzer eines Systems in der Lage, es selber an ihre Umgebungs und Verhaltensweisen anzupassen. Das stellt einen großen Fortschritt gegenüber Systmen dar, die bereits vorab vom Designer auf eine bestimmte Benutzergruppe oder Umgebung zugeschnitten sind.

Schlußendlich hängt die Wahl des zur Merkmalsextraktion und -selektion gewählten Verfahrens also stark vom Problem ab. Gänzlich auf sie zu verzichten wird aber nur bei den wenigsten ubiquitären Systemen möglich sein.

Literatur

1. Paul Duff, Michael McCarthy, Angus Clark, Henk Muller, Cliff Randell, Shahram Izadi, Andy Law, Sarah Pennington, and Richard Swinford. A new method for autocalibrated object tracking. *Proceedings of the Seventh International Conference on Ubiquitous Computing*, pages 123–140, 2005.
2. Matthew Turk and Alex Pentland. Eigenfaces for face detection/recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
3. Volker Baier, Lorenz Mösenlechner, and Matthias Kranz. Gesture classification with hierarchically structured recurrent self-organizing maps. *Fourth International Conference on Networked Sensing Systems*, pages 81–84, 2007.
4. Kanti Mardia and J. T. Kent and J. M. Bibby. Multivariate Analysis (Probability and Mathematical Statistics). *Academic Press*, 1980.
5. Peltarion Corporation. Peltarion synapse. *http://www.peltarion.com/products/products.html* zugegriffen am 24.01.2008
6. MIT Media Lab. Photobook/eigenfaces demo. *http://vismod.media.mit.edu/vismod/demos/facerec/basic.html* zugegriffen am 24.01.2008
7. I.T. Jollife. Principal Component Analysis. *Springer-Verlag Berlin-Heidelberg*, 1986.
8. Martin Berchthold and Kristof Van Laerhoven Real-Time Analysis Of Correlations Between On-Body Sensor Nodes (with Topological Map) *Proceedings of the 2nd*

International Workshop on Wearable and Implantable Body Sensor Networks, pages 27–32, 2005.
9. Teuvo Kohonen. Self-Organizating Maps. *Springer-Verlag Berlin-Heidelberg*, 2001.
10. Kristof Van Laerhoven and Ozan Cakmakci. What shall we teach our pants? *Proceedings of the fourth International Symposium on Wearable Computers*, pages 77–83, 2000.

Fusion
Auflösung durch Masse

Chengchao Qu

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer Till Riedel

allem.qu@gmail.com

Abstract. Zur Erkennung von Aktivitäten und Umgebungskontext kann eine Vielzahl von applikationsspezifischen Sensoren eingesetzt werden. Oft macht es jedoch Sinn, einen sehr genauen Sensor durch viele Sensoren ersetzt werden. Gerade im Bereich des Ubiquitous Computing ist die "Unaufdringlichkeit" der Technologie eine grundlegende Voraussetzung, so dass Technologien wie Kameras keine Option sind. Auch Beschleunigungssensoren können durch viele kleine in Kleidung integrierte Sensoren ersetzt werden, die durch geringe Stromaufnahme und Kosten bestechen. Hier werden Techniken der Fusion von einfachen und unpräzisen Sensoren betrachtet und an Beispielen mögliche Anwendungen erläutert.

1 Einleitung

Der englische Begriff Ubiquitous Computing bezeichnet die Allgegenwärtigkeit der Informationsverarbeitung im Alltag von Menschen. PC wird allmählich verschwinden bzw. durch intelligente Gegenstände ersetzt und integriert. Diese „Mensch-Computer-Interaktion“ soll den Menschen bei seinen Tätigkeiten unmerklich unterstützen. Nach der ersten Ära der zentralen Mainframes, die von vielen Wissenschaftlern bedient wurden, sowie der zweiten Ära der PCs, die jedem Nutzer einen eigenen Computer zuordnete, kann UbiComp als die dritte Computer-Ära bezeichnet werden, in der für jede Person viele Computer eingebettet in der Umgebung vernetzt arbeiten [UbCoWiki]. Deswegen steht heutzutage PC, sowie die Verarbeitungsgeschwindigkeit, die Speicherkapazität, oder die Größe des Rechners immer weniger im Mittelpunkt. Sensoren, und die Daten, die von denen ausgegeben werden, sind eben so wichtig für die Realisierung des Ubiquitous Computing. Sensoren sind schon eine beeinflussende Komponente in neuen Anwendungen geworden. Aber ihre Daten müssen intelligenter benutzt werden, wenn wir ihr echtes Potenzial freisetzen möchten. Um dies zu erzielen, braucht man ein verbessertes Verfahren, damit die Sensoren in Rechnersysteme integriert, und ihre Signale gut interpretiert werden können.

Sensoren sind Geräte, die fähig sind, die physikalischen Stimuli (z.B. Bewegung, Licht oder Temperatur) wahrzunehmen und auf die zu reagieren. Aber in realer Welt, diese Definition kann breiter erweitert. In der Vergangenheit benutzten Kumpel einen

Vogel, um sich vor giftigen Gasen früher zu warnen. Sogar Menschen können als Sensoren betrachtet werden.

Die Beobachtung und Reagierung von Sensorzustand spielen dann eine zentrale Rolle, weil die Sensorsignale nutzlos sind, wenn sie nicht richtig beobachtet oder auf die keine entsprechende Reaktion genommen werden. Wie man Geräte baut, die die Ausführung aller dieser Ubiquitous Computing Aufgaben ohne menschliche Beaufsichtigung ermöglichen, ist heute ein großes Thema.

Viele Anwendungen in Ubiquitous Computing nutzen nicht direkt die Ausgaben der Sensoren, sondern erst nach Klassifikation der gespürten Information. Solche Abbildungen von Rohdaten zu klassifizierten Daten können sehr verschieden sein. Manche sind einfach (z.B. Wert aus Thermometer klassifiziert in „warm“ oder „kalt“), aber die meisten sind komplizierter (z.B. Identifizierung eines Objekts von Kamerabild). Normalerweise funktionieren diese Algorithmen durch eine Trainings-/Lernphase. Benutzer und Sensoren nehmen gleichzeitig die Welt wahr, dann haben die Benutzer die Möglichkeit, mit ihrer eigenen Auffassung das Weltmodell in dem System zu annotieren. Ein interessanterer und sehr flexibler Typ der Klassifikation von Sensordaten ist ein Schritt weiter: inkrementelles Lernen. Dadurch ermöglicht es das System, jederzeit neue Kontexte lernen zu können [Laer04].

Man könnte sich so überlegen, dass es ausreichend wäre, wenn man einen so genannten Smart-Sensor (Ein Smart-Sensor ist ein Sensor, der neben der eigentlichen Messgrößenerfassung auch die komplette Signalaufbereitung und Signalverarbeitung in einem Gehäuse vereinigt [SmSeWiki].) baut, aber in der Tat ist es nicht so einfach. In vielen Fällen, egal, wie funktionsmächtig ein einziger Smart-Sensor ist, sind die Rohdaten kaum richtig zu klassifizieren. Durch folgendes Beispiel ist der Grund dafür anschaulicher [Laer04].

Ein Student möchte ein tragbares System bauen, das kontrolliert, wann er an einem Tag sitzt, und wann er senkrecht steht. Er glaubt, dass ein Orientierungssensor reicht aus. Der gibt den Wert „Stehend“ aus, wenn der Student steht, und den Wert „Sitzend“ aus, wenn er sitzt, und setzt den Sensor über sein Knie. Abbildung 1 zeigt seine Idee.

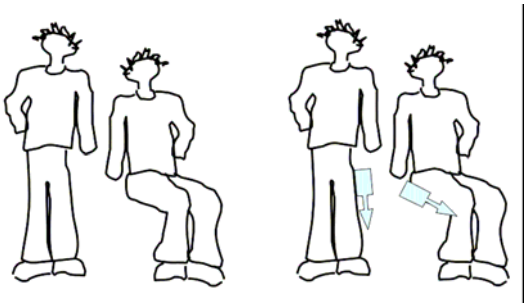


Abb. 1. Beispiel von wie der Status von Benutzer zu entdecken ist [Laer04]

Abbildung 2 zeigt die Schwäche von diesem Ansatz klar an, wenn der Student wie folgendes versucht: Um seinen Sensor zu umspielen, besucht er eine Kneipe. Er steht bei der Bar und setzt seinen Fuß in eine höhere Position, was führt dazu, dass der Sensor sich horizontal orientiert und irrtümlicherweise den Wert „Sitzend“ ausgibt (Abbildung 2a).

Ähnliche Fehler können auch auftreten, wenn sich der Student setzt, aber mit dem Fuß baumelnd. Die Ausgabe ist dann „Stehend“ (Abbildung 2b). Was noch schlimmer ist, wenn er weder steht noch sitzt, z.B. sich niederlegt, gibt der Sensor immer den falschen Wert aus (Abbildung 2c).

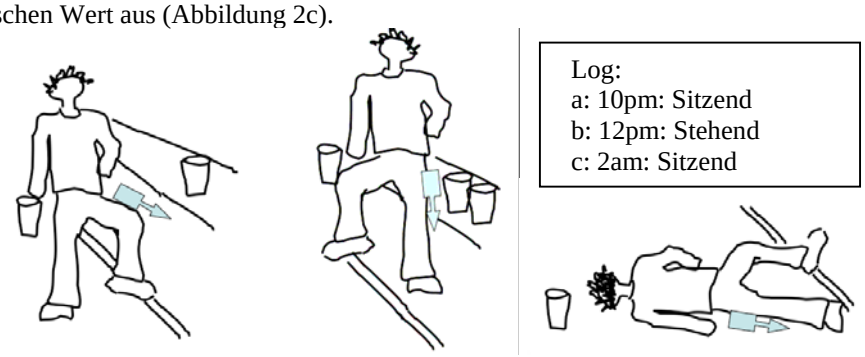


Abb. 2. Einige Beispiele für falsch erkannte Situationen [Laer04]

Dass wir den richtigen Sensor und Algorithmus für Verarbeitung der gespürten Daten haben ist nicht genug, um das System wie erwünscht funktionieren zu lassen. Es ist hier zu betonen, dass solche Fehler nicht durch eine Erhöhung der Empfindlichkeit des Sensors oder eine Verbesserung des Algorithmus für Maschinelles Lernen erhoben werden können. Daraus folgen wir, dass diese Probleme nicht nur in Positionssensor oder Orientierungssensor entstehen. Die Wichtigkeit der Kombinierung von Sensoren tritt dann in den Vordergrund. Z.B. verbessert sich die Performance drastisch, wenn ein gleichartiger Sensor ins andere Bein hinzufügt wird. Aber damit kommen eine neue Herausforderung vor: wie sind die Daten von allen Sensoren zu verknüpfen?

Multi-Sensor Data Fusion bezeichnet die Zusammenführung und Aufbereitung von bruchstückhaften und teilweise widersprüchlichen Sensordaten in ein homogenes, für den Menschen verständliches Gesamtbild der aktuellen Situation [MuSeWiki]. Einfach gesagt, Multi-Sensor Data Fusion verschmelzt Daten aus verschiedenen Sensoren und damit zusammenhängende Information, was ein besseres Ergebnis hat, als was ein Einzelsensor erzielen kann. Der Begriff entsteht in unserem Leben tatsächlich schon lange, ein übliches Beispiel ist Menschen und Tier. Nach tausendjähriger Evolution haben wir die Fähigkeit entwickelt, mit mehreren Sinnen um zu überleben. Beispielweise ist die Kombination von Sicht, Gehör, Geruchsinn, Geschmacksinn und Berührung viel zuverlässiger als mit nur einem. Die Benutzung von mehreren Sensoren hat folgende Vorteile [Laer04]:

- **Billig**. Die kleinen einfachen Sensoren brauchen generell weniger Ressourcen und kosten damit weniger als ein normales Einzel-Sensor-System.
- **Robust**. Falls ein Sensor fällt aus, erfassen andere Sensoren noch ungestört die kontextrelevanten Informationen wegen der Redundanz der Sensoren. Das macht eine niedrigere Fehlerwahrscheinlichkeit möglich. Aus statistischer Sicht, das gleiche Ergebnis von einer Messung durch N unabhängige Sensoren kann nur dann erreicht werden, wenn ein einziger Sensor N mal misst [HaLl01].

- **Verteilt**. Da die Sensoren, die benutzt werden, sehr klein sind, können sie ruhig in einem größeren Gebiet verteilt, oder sogar in Kleidung versteckt werden. Z.B. 2 Winkelsensoren, die die Orientierungswinkel messen, können zusammen die Position eines Objektes bestimmen durch Triangulation [HaLl01].
- **Flexibel**. Das Reichtum und die Komplexität der erkennbaren Kontexte sind direkt von der Anzahl, Position und Art der Sensoren abhängig. Hinzufügen, Bewegen, oder Verbessern der Sensoren steigert dann die Performance des Systems.

In den letzten Jahren hat Multi-Sensor Data Fusion eine große Rolle in militärischen und unmilitärischen Anwendungen gespielt, in Department of Defense (DoD) Gebiet z.B. automatische Zielerkennung, Lenksystem für autonome Fahrzeuge, Schlachtfeldüberwachung und Systeme für automatische Bedrohungserkennung, und in non-DoD Gebiet z.B. Überwachung für komplexe Maschinen, medizinische Diagnose und intelligente Gebäude.

2 Multi-Sensor Data Fusion

In diesem Kapitel werden Hierarchie, Architektur, Prozessmodell und Algorithmen vorgestellt. Es ist auf den Papieren [HaLl97] und [HaLl01] basiert.

2.1 Fusion Level

Beobachtete Daten aus Sensoren können kombiniert, oder fusiert werden bei verschiedenen Fusion-Levels, von Rohdatenlevel zu State-Vektor-Level, oder Decision-Level [HaLl97].

- **Data-Level**. Rohe Sensordaten können direkt kombiniert, wenn die Sensordaten gleich groß sind (z.B. wenn die Sensoren das gleiche physikalische Phänomen messen beispielweise zwei visuelle Imagesensoren oder zwei akustische Sensoren) und Rauschen durch angemessene Algorithmen entfernt (Kalibrierung) wird. Technik für rohe Datenfusion typischerweise besteht aus klassischen Erkennungs- und Bewertungsmethoden. Auf der anderen Seite, wenn die Sensordaten nicht gleich groß sind, sind sie nur bei Feature-/State-Vektor-Level oder Entscheidungslevel zu fusieren.
- **Feature-Level**. Feature-Level-Fusion ist die Extraktion von repräsentativen Features aus Sensordaten. Features sind aus unterschiedlichen Sensorbeobachtungen extrahiert, und dann in einen Feature-Vektor formiert, der als Eingabe der folgenden Mustererkennungsphase gilt. Ein Beispiel dafür ist die Gesichtserkennung. Alle Merkmale des Gesichts werden in einem Vektor gespeichert, um ein Bild von einer Person zu repräsentieren. Algorithmen dazu sind neurales Netz, Clustering-Algorithmus oder Schablonenmethoden.

- **Decision-Level**. Decision-Level-Fusion ist Fusion von Sensorinformationen, die schon eine Vorentscheidung eines Objekts hat. Die Methoden für Decision-Level-Fusion sind typischerweise gewichtete Entscheidung (voting), Bayes'sche Statistik, und Evidenztheorie von Dempster und Shafer.

2.2 Prozessmodell

Eines der historischen Probleme von Datenfusion ist Mangel an einheitlicher Terminologie. Sogar in militärischen Anwendungen sind Definitionen verschieden für Grundbegriff z.B. Korrelation und Datenfusion. Dies führt zu Schwierigkeiten bei der Kommunikation von Entwicklern. Dann im Jahr 1986 wurde die Joint Directors of Laboratories (JDL) gegründet, wovon ein Prozessmodell und ein Datenfusion-Lexikon eingeführt wurden. Das höchste Niveau von JDL Prozessmodell ist in Abbildung 3 angezeigt.

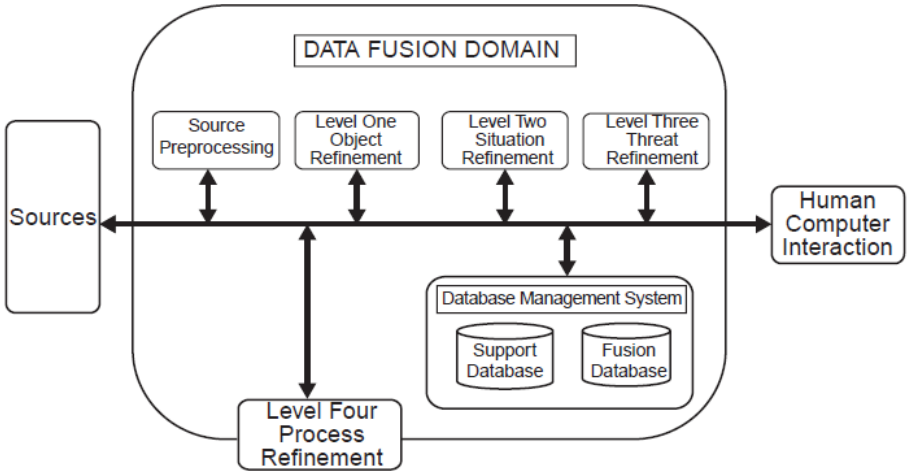


Abb. 3. JDL Prozessmodell für Datenfusion [HaLl01]

- **Sources of Information**. Quellen verfügbar als Eingabe: 1) lokale Sensoren in Verbindung mit einem Datenfusionssystem, 2) verteilte Sensoren verknüpft elektronisch mit einem Fusionssystem, und 3) andere Daten beispielweise Referenzinformation, geographische Information, usw.
- **Human Computer Interaction (HCI)**. HCI ermöglicht Menscheneingabe wie Befehl, Informationsanfrage, Reporte von menschlichen Operatoren, usw. Es dient noch zur Warnung an Menschen vor Aufmerksamkeit, und zur Beseitigung menschlicher kognitiver Begrenzung (z.B. Schwierigkeit in Verarbeitung von negativer Information).
- **Source Preprocessing (Process Assignment)**. Ein Anfangsprozess, der Daten passenden Prozessen zuweist, und Vorabsiebung von Daten ausführt (z.B. Orts- und Attributsdaten an Level 1, Warnungen an Level 3, usw.).

- **Level 1 Processing (Object Refinement)**. Dieser Prozess zielt auf die Kombinierung von Sensordaten, um die zuverlässigste und akkurateste Schätzung zu bekommen.
- **Level 2 Processing (Situation Refinement)**. Dieser Prozess versucht das aktuelle Verhältnis zweier Objekte in Kontext von ihrer Umgebung zu beschreiben.
- **Level 3 Processing (Threat Refinement)**. Dieser Prozess bildet die aktuelle Situation auf die Zukunft ab. Der Threat Refinement im militärischen Sinne geht um eine Schlussfolgerung über Bedrohung von Feind, und Chance für Operationen.
- **Level 4 Processing (Process Refinement)**. Dieser Prozess ist ein Meta-Prozess, der die anderen Data-Fusion-Prozesse kontrolliert, um die Echtzeitleistung des Systems zu schätzen und verbessern.
- **Data Management**. Datenbank Management ist am meisten benutzte Funktion für Sensorfusion wegen der Vielfältigkeit von Daten in Gebrauch, sowie Bedarf an Datenwiedergewinnung, Speicherung, Archivierung, Kompression, relationalen Anfragen, und Datenschutz.

Das JDL Modell ist generisch. Das heißt, obwohl es original für militärische Anwendungen entwickelt wird, ist aber sicherlich auf unmilitärisches Gebiet anwendbar. Z.B. kann Level 3 Threat Refinement in Verbindung mit einer Vorhersage für potenzielle systematische mechanische Fehler gebracht werden.

2.3 Architektur

Wie schon gesagt in Abschnitt 2.1, es gibt drei Alternativen von Sensorfusion: 1) direkte Fusion von rohen beobachteten Sensordaten (so genannte Data-Level-Fusion), 2) Fusion von Feature-Vektoren (In diesem Fall, ein Feature-Vektor ist eine optimale Schätzung basiert auf einzelnen Sensormessungen.), und 3) Fusion von High-Level-Decision jedes einzelnen Sensors. Die Architekturen sind in Abbildung 4 veranschaulicht.

Die erste Architektur (Abbildung 4a) führt Data-Level-Fusion aus. Die rohen Daten, die von jedem Sensor einzeln beobachtet werden, sind fusiert. Darauf gefolgt ist der Entscheidungsprozess. Der erfolgt normalerweise durch eine Extraktion von Feature-Vektor aus den fusierten Daten, und dann eine Umformung von dem Feature-Vektor zu einer Entscheidung. Um die rohen Daten zu fusieren, müssen die originalen Sensordaten einander entsprechen (z.B. müssen die Beobachtungen von gleichen oder ähnlichen physikalischen Größe).

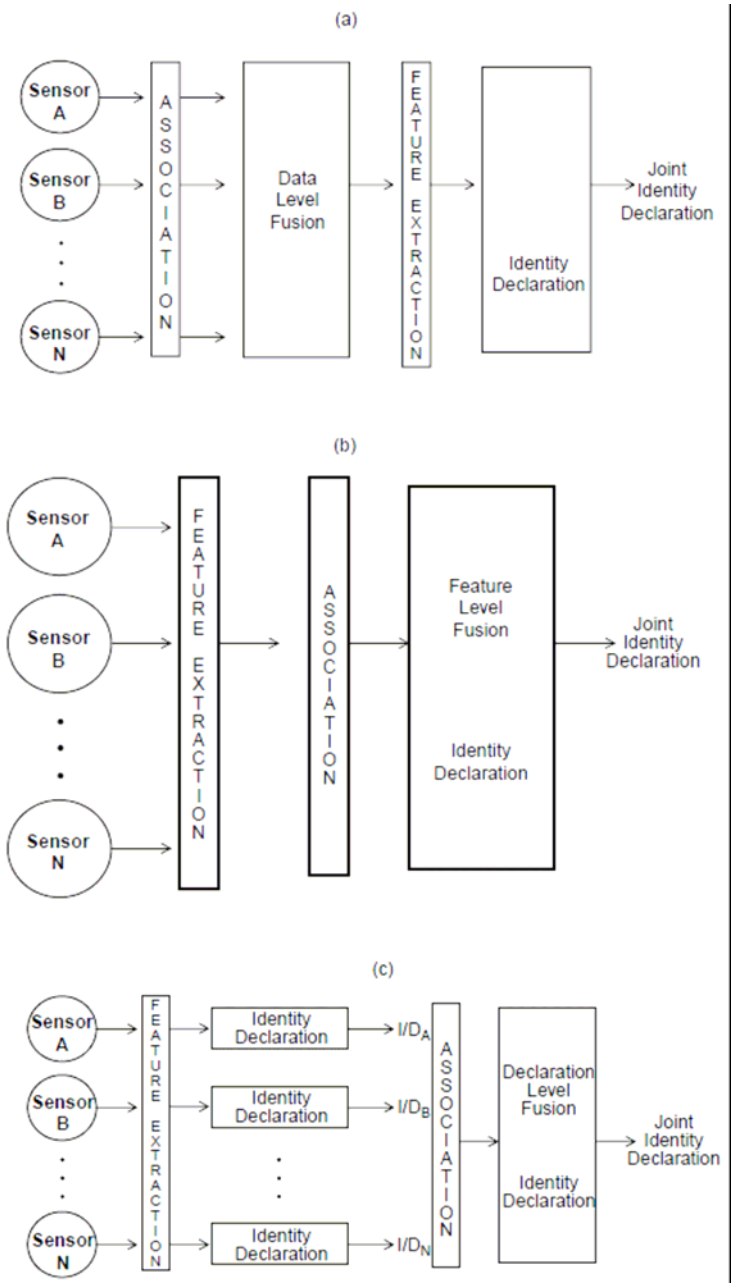


Abb. 4. (a) Direkte Fusion von Sensordaten (b) Fusion von Feature-Vektoren (c) Fusion von High-Level-Decision jedes einzelnen Sensors [HaLl01]

Die zweite Architektur (Abbildung 4b) ist eine Feature-Level-Fusion. Die Ausgabe ist eine fusierte Entscheidung basiert auf den verknüpften Feature-Vektoren aus allen Sensoren. Zu beachten ist, dass die Funktionen von Data Alignment und Assoziation/Korrelation müssen trotzdem früher als die Verbindung der Feature-Vektoren aus einzelnen Sensoren zu einem einzigen größeren Feature-Vektor ausgeführt werden.

Die dritte Architektur (Abbildung 4c) ist eine Decision-Level-Fusion. In diesem Fall entscheidet sich jeder Sensor nur nach seinen einzelnen Daten. Dann werden die allen Entscheidungen durch Algorithmen für Decision-Level-Fusion fusiert (siehe Abschnitt 2.1). Wie andere Architekturen, Data Assoziation und Korrelation sind noch genötigt für die Gewährleistung, dass die Daten bezogen auf das gleiche physikalische Objekt fusiert sind.

Es gibt natürlich viele anderen Modellierungen und Architekturen für Multi-Sensor Data Fusion. Außerdem ist Kategorisierung nach Sensorarten (homogen oder heterogen) oder Fusionsarten (zentralisiert oder dezentralisiert/verteilt) auch möglich und verfügt über verschiedene Arten von Algorithmen. Für die Kalibrierung von rohen Sensordaten existiert eine Vielzahl von Methoden, die in der Datenverarbeitungsphase liegen. Eine passende Architektur und ein passender Algorithmus sollen nach jeglicher Art gewählt werden. In dem nächsten Kapitel werden zwei Beispielanwendungen für Sensorfusion vorgestellt.

3 Beispielanwendungen

Am Beispiel der Anwendungen werden Konzepte der Multisensorfusion erläutert, die verschiedene Ansätze bei Rohdatenverarbeitung, Architekturen und Algorithmen haben, aber am Ende ein beinahe perfektes Ergebnis ausgeben.

3.1 Fuzzy-Validation und Fusion für drahtlose Sensornetzwerke

In [Wen04] wird eine Data-Level-Fusion von einem homogenen Lichtsensor-Netzwerk für intelligente Gebäude ausgeführt mit dem Algorithmus Mote-FVF (Gauß'sche Korrelation und Fuzzy-Logik für Validation/Kalibrierung und Sensing), der die rohen Sensorausgaben mit einem so genannten Vorhersagewert kalibriert und fusiert, der einfach Fuzzy-Logik benutzt.

Die Entwicklung von intelligenten Gebäuden ist ein wichtigstes Thema in non-DoD Gebiet. Viel Energiekonsum kann eingespart, wenn solche Kontrollsysteme für große Bürogebäude eingeführt werden. In diesem Fall ist die Beleuchtungsstärke an allen Arbeitsplätzen unterschiedlich, deshalb ist ein einziger Sensor nicht fähig, die Beleuchtung für individuellen Arbeitsplatz und die gesamte Umgebung zu messen.

Obwohl verteilte Netzwerksensoren, so genannte Motes, kleiner, flexibler und billiger als andere Alternativen sind, haben sie aber auch schwächere individuelle Zuverlässigkeit. Da nach höherer Genauigkeit jeder jagt, ist Sensor Validation und Fusion Algorithmus sehr verbreitet in den letzten Jahren, der relevante Information aus enorm viel Daten extrahiert und Fehler am Anfang erkennt. Mote-FVF benutzt Fuzzy-Logik, um die Korrelation von Sensorausgaben zu definieren, den

Zuversichtswert (confidence value) zu bestimmen und einen gewichteten Durchschnitt zu fusieren. Dieser Algorithmus reagiert auf unerwartete Wechsel rechtzeitig, sofern falsche Sensorausgeben gesiebt werden.

Sensorart
Die homogenen Photosensoren am Crossbow MICA Multi-Sensor Board [CrTe03a], der mit dem MICA Mote Processor [CrTe03b] programmierbar ist und Beleuchtungsstärke spürt, sind verteilt auf einem Prüfstand.

Algorithmus
Der Fuzzy Validation und Fusion (Mote-FVF) Algorithmus benutzt eine Zeitreihe-Vorhersagefunktion, dynamische Validationskurve, die durch Sensorcharakteristiken bestimmt wird, und ein Fusionsschema, das den Zuversichtswert (confidence value) für die Messungen, den Vorhersagewert und den Systemzustand benutzt [PrMo01]. Es gibt drei Einheiten in diesem Algorithmus: Validation, Fusion und Vorhersage.

Validationseinheit
Der Validationsteil der Algorithmus validiert jeden Eingabewert mit einem Zuversichtswert, der durch eine dynamische (Gauß'sche) Validationskurve bestimmt. Für diese Kurve sind die spezifischen Sensorcharakteristiken, der Vorhersagewert, die Korrelation zwischen Eingabewerten, und die physikalische Einschränkung des Sensorwerts relevant. Der Sensorwert gilt nur im Bereich des Validationsgatters. Außerhalb dieses Bereichs wird der Sensorausgabe der Zuversichtswert „0" zugeteilt und der maximale Zuversichtswert ist „1", wenn die Sensorausgabe mit dem Wert in Mitte des Gatters, das von dem Vorhersagewert und der Korrelation aller Sensorausgaben abhängig ist, entspricht. Diese Korrelation zwischen Sensorausgaben kann durch Gauß'sche Korrelationskurve (Abbildung 5) erzielt werden (geht aber auch mit anderen Ansätzen z.B. Medianwert).

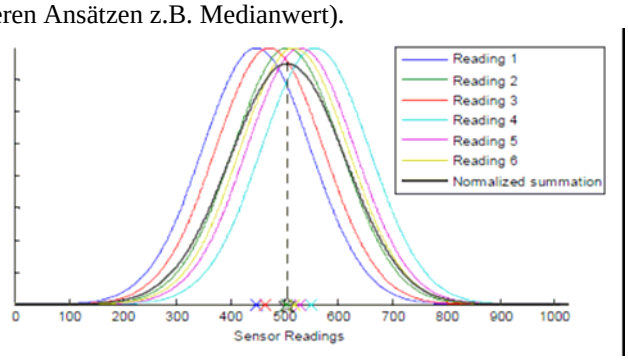


Abb. 5. Gauß'sche Korrelationskurve [Wen04]

Als erster Schritt werden die offensichtlich falschen Sensorausgaben gegen die Einschränkung der Sensoren entfernt. Eine Gauß'sche Funktion (Reading 1 bis Reading 6 in Abbildung 5) wird auf alle übriggebliebenen Sensorausgaben, die eine fein abgestimmte normale Abweichung haben, zentriert. Dann wird die normalisierte

Summierung (Summe durch Anzahl der Messungen) der Gauß'schen Funktionen berechnet.

Für die Validationskurve wird hier die stückweise Gauß'sche Kurve benutzt. Der Parameter für linke und rechte Kurve ist separat gewählt. Und eine Normalisierung ist erforderlich, um die Zuversichtswerte σ zwischen 0 und 1 zu skalieren [GoAg99].

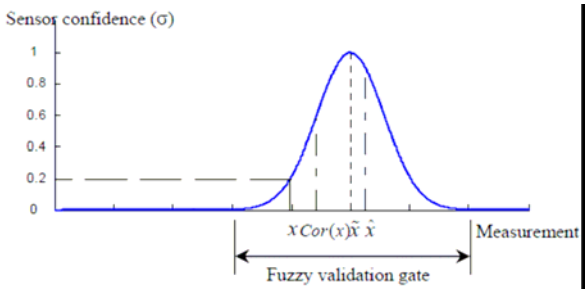


Abb. 6. Unscharfe (Fuzzy) dynamische Validationskurve [Wen04]

α ist durch die Fuzzyregel eingestellt. Es ist ein Tradeoff zwischen dem Vorhersagewert und den meisten Sensorausgaben. Es steht $Cor(x)$ für von Gauß'scher Korrelationskurve bestimmte normalisierte Summierung aller vernünftigen Sensorausgaben, und $Var(x)$ für die Differenz zwischen $Cor(x)$ und dem Vorhersagewert $\hat{x}$. Die Anfangsbedingung der Fuzzyregel ist, dass α gleich wie $\hat{x}$ gesetzt ist,

`IF` $Var(x)$ `klein THEN bewege` α `nach` $Cor(x)$ `eine kleine Menge,`

`IF` $Var(x)$ `mittelgroß THEN bewege` α `nach` $Cor(x)$ `eine mittelgroße Menge,`

`IF` $Var(x)$ `groß THEN bewege` α `nach` $Cor(x)$ `eine große Menge.`

Abbildung 6 zeigt wie α dazwischen schwankt. Die charakteristische Funktion der Fuzzyregel ist in einfachster Form mit Dreieck und maximaler Überlappung (siehe Abbildung 7). m_{var} und m_{mov} jeweils für Fuzzification und Defuzzification sollten eingestellt werden.

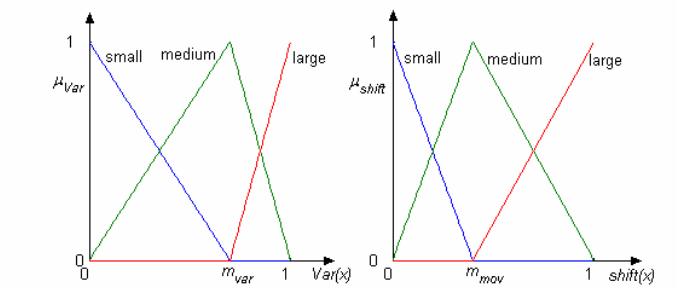


Abb. 7. Charakteristische Funktion für Fuzzyregel, die die Mitte der Validationskurve bestimmt [Wen04]

Fuzzy-Logik ist eine Verallgemeinerung der zweiwertigen Booleschen Logik um einen weiteren unscharfen (engl. fuzzy) Wahrheitswert, der zwischen wahr „1“ der falsch „0“ liegt. Beispielsweise kann der Wahrheitswert den Wert „0.5“ annehmen, so dass damit auch unscharfe Angaben wie "ein bisschen", "ziemlich" oder "stark" mathematisch behandelt werden können [FuLoWiki]. Abbildung 7 gibt eine lineare Fuzzyfunktion an. Für $Var(x) = \frac{1}{4} m_{var}$ ist es 0.75 „small“ und 0.25 „medium“. Dann ist μ_{var} für „small“ und „medium“ entsprechend direkt aus dem Dreieck abzulesen. Für den Fall Defuzzification von μ_{shift} ist es ähnlich.

Fusionseinheit

Die Struktur der Fusionseinheit passt genau zu der Data-Level-Fusion (siehe Abschnitt 2.3). Eine Skizze davon ist in Abbildung 8 zu finden.

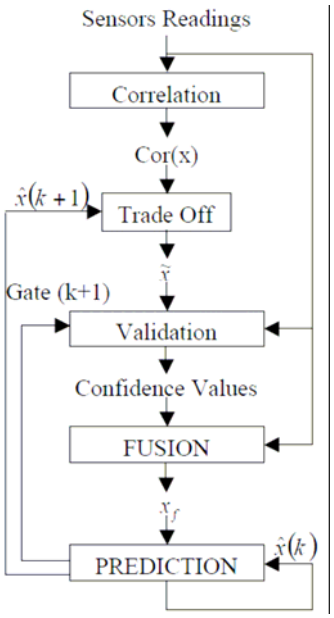


Abb. 8. Skizze des Mote-FVF Algorithmus

Fusion des Mote-FVF Algorithmus ist ein gewichteter Durchschnitt von Sensorausgaben und dem Vorhersagewert,

$$x_f = \frac{\sum_1^n z_i \sigma(z_i) + \frac{\alpha \hat{x}}{\omega}}{\sum_1^n \sigma(z_i) + \frac{\alpha}{\omega}}.$$

Der Skalierungsfaktor ω beinhaltet einen Teil des Vorhersagewerts um die Unstabilität des Systems zu verhindern, dass keine gültige Messung vorhanden ist nach dem Validationsprozess. Darüber hinaus hält der Algorithmus in Stand die Robustheit für einen temporären Sensorfehler. ω ist typischerweise groß gewählt, damit der Vorhersagewert den Fusionswert nicht dominiert. α tritt sowohl in der Fusionseinheit als auch in der Vorhersageeinheit auf. Für einen stabilen Systemzustand wird α groß gesetzt, da in solcher Situation eine große Schwankung in Sensorausgaben möglicherweise wegen Rauschen verursacht werden kann; für einen instabilen Systemzustand, jedoch, wird α klein gesetzt, da in dieser Situation der Vorhersagewert wenig zu tun mit der Wirklichkeit hat. α wird auch durch die Fuzzyregel bestimmt [KhKe92]:

```
IF Schwankung der Sensorausgaben klein THEN α groß,

IF Schwankung der Sensorausgaben mittelgroß THEN α
mittelgroß,
```

`IF Schwankung der Sensorausgaben groß THEN` α `klein.`

Dreieckfunktion mit maximaler Überlappung ist hier wieder geeignet für die charakteristische Funktion der Fuzzyregel (Abbildung 9). m_e ist für Fuzzification und m_α ist für Defuzzification.

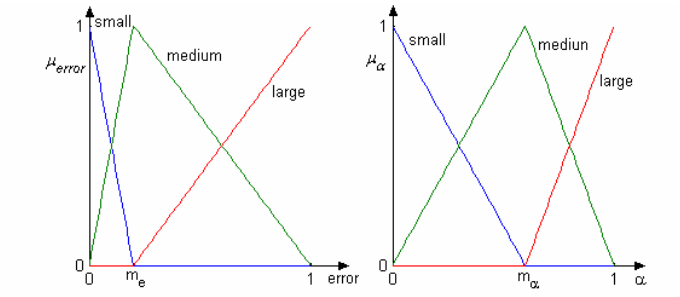


Abb. 9. Charakteristische Funktion der Fuzzyregel für α [Wen04]

Vorhersageeinheit
Der Vorhersagewert für den nächsten Schritt wird durch eine Zeitreihe-Vorhersagefunktion generiert. Ein Tradeoff zwischen Änderungssensitivität, Glätte, Stabilität, und Nachteile der Vorhersagefunktion muss berücksichtigt werden. Die Form ist $\hat{x}(k+1) = \alpha\hat{x}(k) + (1-\alpha)x_f(k)$. $\hat{x}(k)$ ist der Vorhersagewert des aktuellen Schritts und $x_f(k)$ ist der aktuelle Fusionswert [GoAg96] [GoAg99].

Leistungstest
6 Motes in einer 3×2 Matrix werden die Beleuchtungsstärke auf dem Prüfstand testen. Es kann eine heftige Änderung in Beleuchtungsstärke sein. Z.B. wird das Licht plötzlich ein- oder ausgeschaltet. Hier lässt sich der Algorithmus in diskontinuierlicher Änderung testen. Ein großer Bereich der Beleuchtungsstärke, von 0 Lux (total dunkel) zu 1000 Lux (sehr hell), wird getestet. Nullwertiger Zustand kann als Sensorfehler oder totale Dunkelheit interpretiert werden. Die beiden sollte der Algorithmus gut unterscheiden. Abbildung 10 zeigt das Ergebnis an.

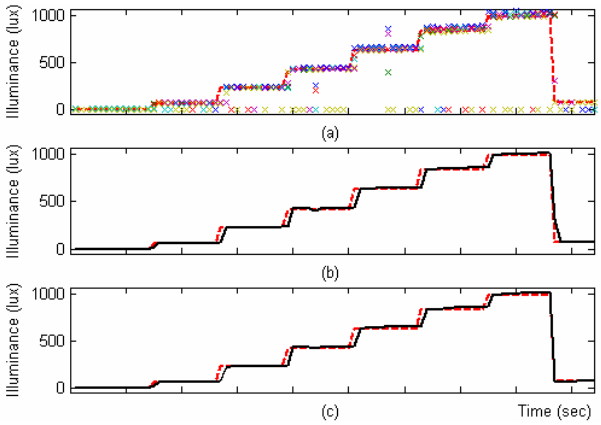


Abb. 10. (a) Rohe Sensorausgaben und echte Beleuchtungsstärke (b) Mote-FVF Algorithmus mit Medianwert (c) Mote-FVF Algorithmus mit Gauß'scher Korrelation [Wen04]

(a) gibt die rohen Sensorausgaben mit Kreuzungen und echte Beleuchtungsstärke mit roter punktierter Linie. Der Ansatz von (b) nutzt direkt den Medianwert aller Sensorausgaben als zu fusionierter Wert statt der Gauß'schen Approximation. Deswegen bei der letzten steil abgestiegenen Phase des Testes ist das Ergebnis nicht so akkurat wie der Ansatz mit Gauß'scher Korrelation, da die reelle Beleuchtung zu nah zu „0" ist. Die beiden funktionieren ziemlich gut. Bei den Änderungen verzögert sich der Algorithmus eine Weile, bis er mit ändert, was auf die Vorhersageeinheit zurückzuführen ist. Eben auch dank dieser Einheit werden Rauschen in der Mitte von (a) gut eliminiert.

3.2 Navigation im Haus mit unkalibrierten heterogenen Sensoren

Eine Data-Level-Fusion mit Kalibrierung, die im obigen Beispiel vorgestellt wurde, sieht gut und einfach aus. Es ist bekannt, die Data-Level-Fusion ist am akkuratesten [HaLl97], dann wieso noch Feature-Level-Fusion? In dem folgenden Abschnitt wird eine Feature-Level-Fusion von heterogenen Sensoren für die Navigation im Haus ausgeführt, bei der unterschiedliche Sensorarten in Anspruch genommen werden und die nicht als zentralisierte Fusion geeignet ist. Dieser Abschnitt ist basiert auf [GoLe99].

Da Global Positioning Systems (GPS) nicht in einem Gebäude funktionieren, werden in diesem Papier Sensoren als Alternativen in Rücksicht genommen, die den Ort durch Beschleunigung, Magnetfeld, Temperatur und Beleuchtungsstärke bestimmen. Der anfängliche Versuch mit direkter Nutzung von rohen Sensordaten läuft sehr schlecht mit einer Fehlerrate von fast 50%. Dann wird ein Data-Cooking-Modul eingeführt, das stattdessen ein High-Level-Feature berechnet. Dies führt zu einer erheblichen Performanceerhöhung auf Fehlerrate von 2%.

Sensorart
Neun Sensoren von vier Arten (Beschleunigungssensor, Magnetometer, Temperatursensor und Lichtsensor) werden eingesetzt. Tabelle 1 listet die Sensoren in Gebrauch auf.

Sensor	Description	Units
Left X acc.	Acceleration to user's right (measured at user's left hip)	G
Left Y acc.	Acceleration forward (measured at user's left hip)	G
Right X acc.	Acceleration to user's right (measured at user's right hip)	G
Right Z acc.	Acceleration upward (measured at user's tight hip)	G
Comp. X	Component of magnetic field pointing to user's right	Gauss
Comp. Y	Component of magnetic field pointing forward	Gauss
Comp. Z	Component of magnetic field pointing upward	Gauss
Temperature	Ambient room temperature	Degrees Fahrenheit
Lights	Strength of 60 Hz component in overhead lights	Volts

Tabelle 1. Sensoren und ihre kanonischen Einheiten der Messungen [GoLe99]

Algorithmus
Der Algorithmus besteht aus 3 Modulen: der Data-Cooking-Modul ist das wichtigste für die Verarbeitung der unkalibrierten Rohdaten. High-Level-Feature wird dann berechnet, z.B. die Varianz von Z-Beschleunigung des Benutzers. Da dieser Modul mehr nachvollziehbar nach den folgenden zwei Modulen ist, wird er am Ende vorgestellt; der Data-Modelling-Modul und Navigationsmodul lernen in der Trainingsphase die Welt, in der sich das Navigationssystem befindet, und danach folgert den Ort des Benutzers im Laufzeit.

Data-Modelling
Das Training funktioniert wie folgendes: der Benutzer folgt die vorbestimmte Routine, drückt einen Knopf beim Eintritt und Verlassen eines Ortes, damit das System über die echte Welt informiert wird. Abbildung 11 zeigt diese Testumgebung mit 5 Orte: Büro (office), Wölbung (arch), Treppe hinauf (upstairs), Treppe hinunter (downstairs) und Tischtennisraum (ping pong room).

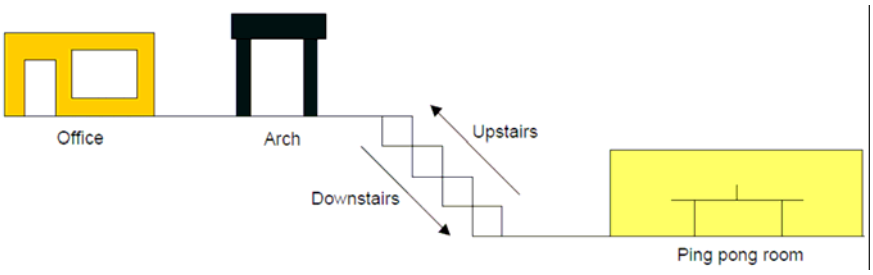


Abb. 11. Testumgebung bestehend aus fünf Orten [GoLe99]

Auf jedes Tupel der Sensorausgaben wird der spezifische Ort abgebildet. Danach konstruiert der Data-Modelling-Modul eine Verteilung, die den Charakteristiken eines Ortes passt. Die echte Welt besteht dann aus einer Menge von möglicher Werten der Sensoren, eine Verteilung für jedes <Sensor, Ort> Paar.

Navigation

Der Algorithmus vereinigt zwei Aspekte von Wissen für die Ermittlung des Ortes des Benutzers. Der erste ist das Wissen über die Verteilung des Signals an jeglichem Ort, was in der Trainingsphase gelernt wird. Der zweite ist das so genannte Koppelnavigationswissen (engl. dead-reckoning, laufende Ortsbestimmung/Ortung eines bewegten Objekts durch Messen der Bewegungsrichtung, der Geschwindigkeit und der Zeit [KoNaWiki]). Es ist ein iterativer, Markov-Modell ähnlicher, Zweiphasenalgorithmus. Jedes Mal wenn ein Sensortupel ausgegeben ist, wird die Wahrscheinlichkeit inkrementell updatet.

Der erste Schritt wird durch die Bayes'sche Regel erfolgt. Sei $S = \langle s_1, ..., s_n \rangle$ ein vergrößertes (nach der Data-Cooking-Phase) Tupel, für jeden Ort l eine A-posteriori-Wahrscheinlichkeit, dass der Benutzer jetzt an diesem Ort ist, wird berechnet mit der Annahme, dass die Bedingung unabhängig ist:

$$P(l \mid \langle s_1, ..., s_n \rangle) = (\prod_{1 \le i \le n} P(s_i \mid l)) \frac{P(l)}{P(\langle s_1, ..., s_n \rangle)}.$$

$P(s_i \mid l)$ steht für den i-ten Sensor am Ort l, und wird mit dem in der Trainingsphase gelernten Modell berechnet. Die A-Priori-Wahrscheinlichkeit $P(l)$ wird vor dem Update dem Ort l zugewiesen. $P(\langle s_1, ..., s_n \rangle)$ wird vernachlässigt, die Wahrscheinlichkeiten für alle Orte summieren sich auf 1.

Die zweite Phase hat die Koppelnavigation in Gebrauch. Wenn ein Schritt erkannt ist, bedeutet p_{stay}, dass sich der Benutzer nach dem Schritt noch an dem Ort befindet und wird schätzt durch die Trainingsdaten; $(1 - p_{stay})$ wird gleichmäßig unter alle Orte nebenan aufgeteilt.

Data-Cooking

Diese Data-Cooking-Phase ist eine der wichtigsten in diesem Algorithmus, da ohne dies die Leistung ziemlich schlecht ist (nahezu 50% ohne High-Level-Feature, siehe Tabelle 2).

Das erste Problem kommt aus übertriebenen Unabhängigkeitsannahmen in dem Bayes'schen Update, aufeinanderfolgende Tupeln sind auch unabhängig betrachtet. Deswegen wenn der Benutzer unter Licht für eine Dauer steht, folgert das System, dass er in dem hellsten Raum ist. Aus diesem Grund muss ein Feature hinzugefügt werden, dass es nur ausgelöst, wenn das Licht von „Aus" zu „Ein" geschaltet wird, nicht während das Licht an ist.

Das zweite Problem ist absolute Messung gegen relative Messung. Sensorausgaben wandern oft, z.B. ein Beschleunigungssensor bewegt sich an dem Gürtel, und beeinflusst die Schwerkraftmessung. Deswegen wird ein Feature berechnet, dass der Sensor statt des absoluten Wertes ein Delta (Differenz) zwischen dem aktuellen Wert und dem Durchschnitt einer Dauer.

Das dritte Problem bezieht sich auf den Kontext von Z-Beschleunigungssensor. Es ist beobachtet, wenn der Benutzer die Treppe hinunter geht, steigt die Schwankung in Z-Beschleunigung. Deswegen ist ein Feature von der Varianz der Z-Beschleunigung eingeführt und der Ort „Downstairs" ist dann verlässlich zu unterscheiden. Abbildung 12 veranschaulicht das neue Feature gegen rohes.

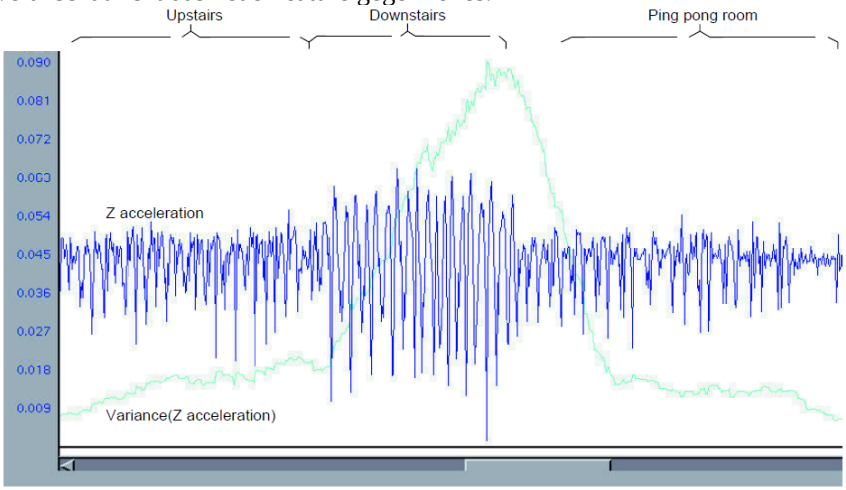


Abb. 12. Feature von Z-Beschleunigung (violett) gegen Varianz der Z-Beschleunigung (hell blau) [GoLe99]

Das letzte Problem ist über das Rauschen der Sensorverteilungen. Z.B. schwankt der Compass Z Sensor immer, aber solches Rauschen weist auf keinen Ort hin. In diesem Algorithmus wird dann einen Schwellwert (restricted range) benutzt, alle Sensorausgaben, die unter diesem Schwellwert liegen, werden gelöscht und damit nicht ins Bayes'sche Update kommen.

Leistungstest
Es ist verständlich, dass das System erst nach einiger Zeit reagiert und eine Entscheidung trifft, nachdem der Benutzer in einen Ort eintritt. Diese Wartezeit heißt „Kopf“ (header). Der „Körper“ (body) ist die übrige Zeit, die der Benutzer danach in dem gleichen Ort verpasst. Es ist klar aus Tabelle 2, dass die Einführung von Data-Cooking erhöht die Leistung beträchtlich.

High-level features	Restricted ranges	Time until first guess	Errors per visit	Error rate
yes	yes	4.8 secs	0.3	2%
yes	no	2.0 secs	2.6	33%
no	yes	3.4 secs	10.1	52%
no	no	2.1 secs	10.5	49%

Tabelle 2. Testergebnisse [GoLe99]

4 Fazit

Die Einführung eines allgemeinen Modells für Sensorfusion erleichtert in gewissem Masse die Systementwickler beim Entwurf. Obwohl bei den zwei hier erwähnten Beispielanwendungen die Architekturen nicht so offensichtlich zu sehen sind, ist eines trotzdem klar, dass für unterschiedliche Anwendungen verschiedene Algorithmen zum Einsatz kommen müssen. Für den ersten Fall handelt es sich um ein homogenes Sensornetzwerk, die Ausgaben aller Sensoren sind gleich und können direkt in Benutzung genommen werden. Deshalb ist eine zentralisierte Fusion auf Data-Level besonders geeignet. Sogar sind solche Methoden für homogene Fusion auch auf heterogene Ebene anwendbar, wenn der ganze Prozess in kleine Sensorgruppen aufgeteilt wird und am Ende die bearbeiteten Daten zu dem Zentralprozessor zurückgesendet werden. Dadurch sind die Nachteile der zentralisierten Fusion, z.B. ein hoher Bedarf an Bandbreite für die globale Kommunikation, und starker Overhead (damit gibt es eine theoretische Grenze von Anzahl der Sensoren in zentralisierter Fusion), zu vermeiden.

Eine große Herausforderung in Sensorfusion, oder noch allgemeiner, in Sensortechnologie, ist ein „weiser“ Gebrauch der rohen Daten. Wie in der zweiten Beispielanwendung gezeigt wurde, läuft das System sehr schlecht ohne diese Data-Cooking-Phase (Kalibrierung). Direkte Sensorausgaben neigen zu Fehler aufgrund von Ausfall der Sensoren, innerem oder äußerem Rauschen usw. Diese Kalibrierungsphase gibt High-Level-Feature aus, das auch unter Zustand verstanden werden kann, dennoch sinnlos für homogenen Sensorausgaben wie in der ersten Beispielanwendung Beleuchtungsstärke, addiert aber kein neues Wissen zu dem System, sondern gestaltet rohe Sensordaten in eine Form um, die Algorithmen für Maschinelles Lernen gut verwenden können. Nur dann wird die Macht der Sensorfusion freigesetzt. Kurz gesagt, für die drei Fusion-Levels ist eine große Menge von Algorithmen vorhanden, aber nur solche, die am meisten passend sind, gibt gutes Resultat aus (eine komplette Liste von Algorithmen, die zu welchen Fusion-Level passen, wird in [HaLl97] aufgeschrieben). Somit ist ein kontextbewusstes System

durch Integration der Information aus vielen kleinen, billigen und verteilten homogenen/heterogenen Sensoren realisierbar.

Sensorfusion, die heute schon im Vordergrund des Ubiquitous Computing steht, wird morgen in unseren täglichen Leben immer mehr an Bedeutung gewinnen.

5 Literatur

[Wen04] Wen, Y.-J., Agogino, A.M., Goebel, K. “Fuzzy Validation and Fusion for Wireless Sensor Networks”. Proc., ASME International Mechanical Engineering Congress, Anaheim, CA. 2004.

[Laer04] Van Laerhoven, K. “The Pervasive Sensor”. Invited Talk, In Ubiquitous Computing Systems, Second International Symposium, UCS 2004, Tokyo, Japan. 2004. Revised Selected Papers, LNCS 3598, Springer 2005, ISBN 3-540-27893-1, pp.1-9.

[GoLe99] Andrew R. Golding, Neal Lesh. “Indoor Navigation Using a Diverse Set of Cheap, Wearable Sensors”. Proc. 3rd IEEE International Symposium on Wearable Computers. 1999. pp. 29-36.

[HaLl97] David L. Hall, James Llinas. “An Introduction to Multisensor Data Fusion”. Proc., IEEE. 1997. Vol. 85, Issue 1, pp. 6 - 23.

[HaLl01] David L. Hall, James Llinas. “Handbook of Multisensor Data Fusion”. CRC Press LLC. 2001. Chapter 1, pp. 18-27.

[CrTe03a] Crossbow Technology. “MTS/MDA Sensor and Data Acquisition Boards User’s Manual”. Rev. B, San Jose, CA. 2003.

[CrTe03b] Crossbow Technology. “MPR- Mote Processor Radio Board, MIB- Mote Interface/Programming Board User’s Manual”. Rev. A, San Jose, CA. 2003.

[PrMo01] Prajitno, P., Mort, N. “A Fuzzy Model-Based Multi-Sensor Data Fusion System”. Proc., SPIE Conference on Sensor Fusion: Architectures, Algorithms, and Applications V, Orlando, FL. 2001. Vol. 4385, pp. 301-312.

[GoAg96] Goebel, K., Agogino, A. M. “An Architecture for Fuzzy Sensor Validation and Fusion for Vehicle Following in Automated Highways”. Proc., 29th International Symposium on Automotive Technology and System/Soft Computing in the Automotive and Transportation Industry, Florence, Italy. 1996. pp. 203-209.

[GoAg99] Goebel, K., Agogino, A. M. “Fuzzy Sensor Fusion for Gas Turbine Power Plants”. Proc,. SPIE Conference on Sensor Fusion: Architectures, Algorithms, and Applications III, Orlando, FL. 1999. Vol. 3719, pp. 52-61.

[KhKe92] Khedkar, P., Keshav, S. “Fuzzy Prediction of Time Series”. Proc., IEEE International Conference on Fuzzy Systems, San Diego, CA. 1992.

[UbCoWiki] Ubiquitous Computing: http://de.wikipedia.org/wiki/Ubiquitous_Computing. 25.12.2007.

[SmSeWiki] Smart-Sensor: http://de.wikipedia.org/wiki/Smart-Sensor. 20.11.2007.

[MuSeWiki] Multi-Sensor Data Fusion: http://de.wikipedia.org/wiki/Multi-Sensor_Data_Fusion. 12.09.2007.

[FuLoWiki] Fuzzy-Logik: http://de.wikipedia.org/wiki/Fuzzy-Logik. 23.01.2008.

[KoNaWiki] Koppelnavigation: http://de.wikipedia.org/wiki/Koppelnavigation. 07.11.2007.

Wie/Was/Wo sind Sensordaten? (von Ontologie bis Darstellung)

Clemens Koller

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer Till Riedel

1 Einleitung

Sensoren spielen heutzutage eine wichtige Rolle in der Steuerung von vielen alltäglichen Abläufen, zum Beispiel zur Verkehrsüberwachung oder für Wettervorhersagen. Ein Großteil dieser Sensoren befindet sich in zueinander inkompatiblen und anwendungsspezifischen Sensorinfrastrukturen, die zudem häufig von verschiedenen kommerziellen Betreibern eingesetzt werden. In den letzten Jahren haben sich dank dem Trend zu überall verfügbarem Internetzugang jedoch auch viele Geräte mit eingebauten Sensoren weit verbreitet, wie zum Beispiel Mobiltelefone mit Kamera und GPS-Empfänger.

Eine Vereinigung all dieser Sensoren über das Internet zu einem Sensorsystem mit einheitlicher und offener Abfragesprache würde neue Anwendungen ermöglichen, die weit über die Möglichkeiten der einzelnen Sensoren hinausgehen. So könnten beispielsweise die Parkplatzsensoren aller Parkhäuser in einer Stadt zusammen mit Bildern ihrer Zufahrtsstraßen von Webcams, aktuellen Verkehrsinformationen und dem Navigationssystem in einem Auto dazu verwendet werden, zum nächsten Parkplatz mit möglichst geringer Wartezeit geleitet zu werden, ohne dabei in einen Stau zu geraten.

Durch eine effiziente Wiederverwendung von bereits vorhandenen Sensoren könnten auch die Kosten und der Aufwand für neue Sensoranwendungen gesenkt werden.

2 Herausforderungen

Bei der Verknüpfung von verschiedenen anwendungsspezifischen Sensorinfrastrukturen zu einem einheitlichen Sensorsystem müssen allerdings einige Herausforderungen überwunden werden, die zum Beispiel durch die Heterogenität der Systemkomponenten und die hohen Anforderungen an die Skalierbarkeit entstehen.

2.1 Sensorbeschreibung

Aufgrund der großen Anzahl verschiedenster Sensoren mit häufig proprietären Zugriffsmöglichkeiten und unterschiedlicher Datenrepräsentation wird eine einheitliche Beschreibung der Sensordaten und -schnittstellen benötigt um Interoperabilität zu ermöglichen. Diese Beschreibung sollte auch leicht erweiterbar

sein damit später Sensoren hinzugefügt werden können, die beim anfänglichen Einsatz des Systems noch vorgesehen waren. Als Format bietet sich die eXtensible Markup Language (XML) [1] an, entweder als eigenständiges Format [2] oder auf Basis von SensorML [3].

Des Weiteren ist auch eine Ontologie, das heißt eine Beschreibung der Beziehungen zwischen den verschiedenen Sensoren als abstrakten Typen, notwendig, beispielsweise mit Hilfe der Web Ontology Language OWL [4] oder auch IEEE SUMO [5]. Dies ermöglicht zum einen die Automatisierung von Prozessen innerhalb des Sensorsystems, zum Anderen wird die Erweiterbarkeit vereinfacht, da neue Sensoren einfach dem abstrakten Typ eines bereits bekannten Sensors zugewiesen werden beziehungsweise Eigenschaften von diesen erben.

2.2 Sensor Discovery

Damit ein Sensorsystem vielseitig einsetzbar ist, benötigt es eine große Anzahl von Sensoren. Es gilt also, das Hinzufügen von Sensoren für den Sensorbetreiber so einfach wie möglich zu gestalten. Problematisch hierbei ist, dass der Sensorbetreiber häufig keinen direkten Nutzen davon hat, seine Sensordaten zur Verfügung zu stellen, da seine eigenen Sensoren bereits alle seine Anforderungen erfüllen, es entstehen ihm aber unter Umständen durchaus Kosten. Das Sensorsystem beziehungsweise seine Nutzer müssen also Anreize zum Hinzufügen von Sensoren bieten.

Eine andere Möglichkeit zur Beschaffung von Sensordaten ist der Zugriff auf öffentliche Sensoren, zum Beispiel Webcams im Internet. Diese automatisch zu finden und klassifizieren ist aufgrund von fehlender standardisierter Struktur und fehlenden Metadaten allerdings schwierig bis unmöglich.

2.3 Datenverwaltung

Durch die große Menge an anfallenden Daten in einem Sensorsystem entstehen bei der Datenhaltung weitere Probleme. Diese unterscheiden sich vom traditionellen Data-Warehouse Modell, bei dem sich die Daten selten ändern, und vom Data-Streaming Modell, das lange andauernde Abfragen auf einem kontinuierlichen Datenstrom ermöglicht. Ein passenderes Modell wäre eine durch Aggregation der Sensoren entstehende ungefähre Sicht auf die Daten, die bei Bedarf verfeinert wird. Auch die Verwendung von Caches bietet sich an um die anfallende Datenmenge weiter zu verringern, wobei sich die Frage stellt, was und wo zwischengespeichert werden sollte. Bei sehr großen Sensorsystemen, zum Beispiel weltweiten Wettersensoren zum Erstellen eines globalen Klimamodells, ist es sinnvoll die Daten möglichst nahe bei den Sensoren zu speichern und nur bei Bedarf zu einem Client zu senden. Aufgrund der großen Datenmengen ist das Speichern aller Daten allerdings nicht möglich, das heißt das Sensorsystem sollte möglichst selbstständig entscheiden können, welche Daten in der Zukunft noch nutzlich sein könnten und deren Speicherung sich somit lohnt.

Ein weiteres Problem entsteht durch die unterschiedlichen Anforderungen der Sensoren. So liefert zum Beispiel ein Temperatursensor in regelmäßigen

Abständen nur einen einzelnen Skalar, während für eine Kamera ein kontinuierlicher Strom von Videodaten bewältigt werden muss.

2.4 Datenabfrage

Damit das Sensorsystem möglichst vielseitig einsetzbar ist, benötigt es eine einfache und flexible Abfragesprache. Hierbei bietet es sich an auf die Ontologie der Sensoren aufzubauen um automatisiert überprüfen zu können, ob und wie eine Anfrage mit den vorhandenen Sensoren und Daten erfüllt werden kann. Die Datenabfrage sollte sowohl den Zugriff auf die gemessenen Sensordaten erlauben als auch auf die Metadaten über den Sensor wie zum Beispiel den Sensortyp, den Zeitstempel der letzten Messung oder den Standort des Sensors. Zum Schutz vor Überlastung einzelner Systeme sollten außerdem Loadbalancing-Verfahren zum Einsatz kommen, und wenn möglich sollten die Daten von bereits bearbeiteten Abfragen wiederverwendet werden.

2.5 Visualisierung

Da sich Sensorsysteme über ein großes geografisches Gebiet erstrecken können, ist eine Visualisierung mithilfe eines Kartendienstes wie Google Maps[1] oder Microsofts Live Maps[2] hilfreich. Die einfache Darstellung der Sensoren als Datenpunkte erweist sich hierbei aber als nicht ausreichend, besser ist eine an den Sensortyp angepasste Darstellung. Für Temperatursensor bietet sich beispielsweise die Einfärbung der Karte anhand von Isothermen an.

Um die Übersicht zu bewahren sollten die Sensordaten des Weiteren wenn möglich sinnvoll aggregiert werden, die Details sollten erst bei näherer Betrachtung sichtbar werden.

2.6 Sonstige Herausforderungen

In einem System, das hauptsächlich zum Sammeln von Daten besteht, ist natürlich auch Datenschutz ein Problem. So muss ein Sensorbetreiber die Möglichkeit haben zu kontrollieren wer auf seine Daten zugreift, und wie diese Daten verwendet werden. Die ist allerdings häufig nicht ausreichend, denn es sollte auch sichergestellt werden, dass der Sensorbetreiber tatsächlich das Recht an seinen Daten besitzt. Zum Beispiel gibt einem eine auf ein Restaurant gerichtete Kamera noch lange nicht das Recht dazu die momentane Wartezeit dieses Restaurants zu veröffentlichen.

3 Systeme

Im Folgenden werden nun einigen Sensorsystemen vorgestellt, und es wird untersucht ob und wie mit den genannten Herausforderungen umgegangen wird.

[1] http://maps.google.de
[2] http://maps.live.de

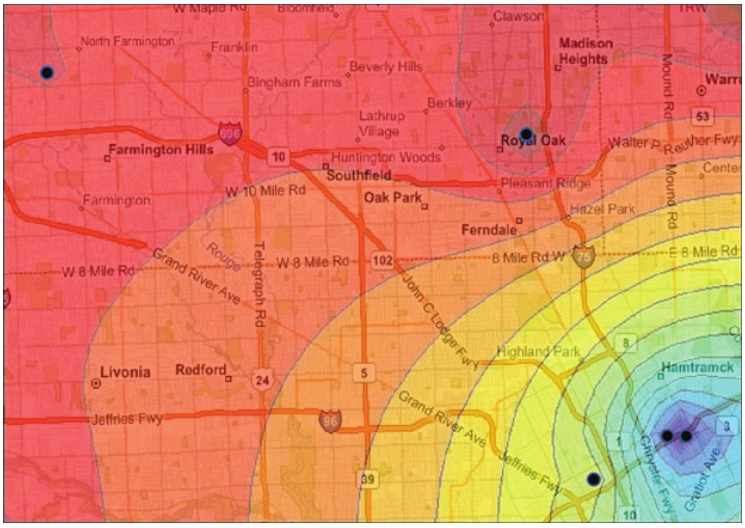


Abbildung 1. Visualisierung von Temperaturdaten. Die schwarzen Punkte zeigen die Sensorstandorte. (Quelle: [6])

3.1 OGC Sensor Web

Im Rahmen der Sensor Web Enablement (SWE) [7] Initiative arbeitet das Open Geospatial Consortium, Inc (OGC)[3] an einem Framework offener Standards zur Nutzung von mit dem Internet verbundenen Sensoren und Sensorsystemen jeglicher Art, von Webcams über Verkehrsüberwachungsanlagen bis hin zu satellitenbasierten Kamerasystemen. Ermöglicht werden soll unter anderem:

- Das Auffinden von Sensorsystemen und Sensorbeobachtungen, die die Anwendungs- beziehungsweise Benutzeranforderungen erfüllen;
- Der Zugriff auf Sensorparameter zur automatischen Verarbeitung und Lokalisierung von Beobachtungen;
- Die Abfrage von Sensordaten in standardisierten Codierungen;
- Die Auftragsbearbeitung in Sensorsystemen.

Um diese Ziele zu erfüllen, gliedert sich SWE in sieben einzelne Standards:

Observations & Measurements Schema (O&M)[8] bietet UML Standardmodelle und XML Schema [9] zur Repräsentation und zum Austausch von Beobachtungsergebnissen, wobei eine Beobachtung als ein Ereignis mit einem Ergebnis, dessen Wert ein bestimmtes Phänomen beschreibt, definiert wird. Es regelt außerdem die Codierung von Sensordaten und vereint die Flexibilität und Erweiterbarkeit von XML mit effizienten Möglichkeiten um große Datenmengen als ASCII- oder Binärblöcke zu speichern. Für die Werte von Messungen können

[3] http://www.opengeospatial.org

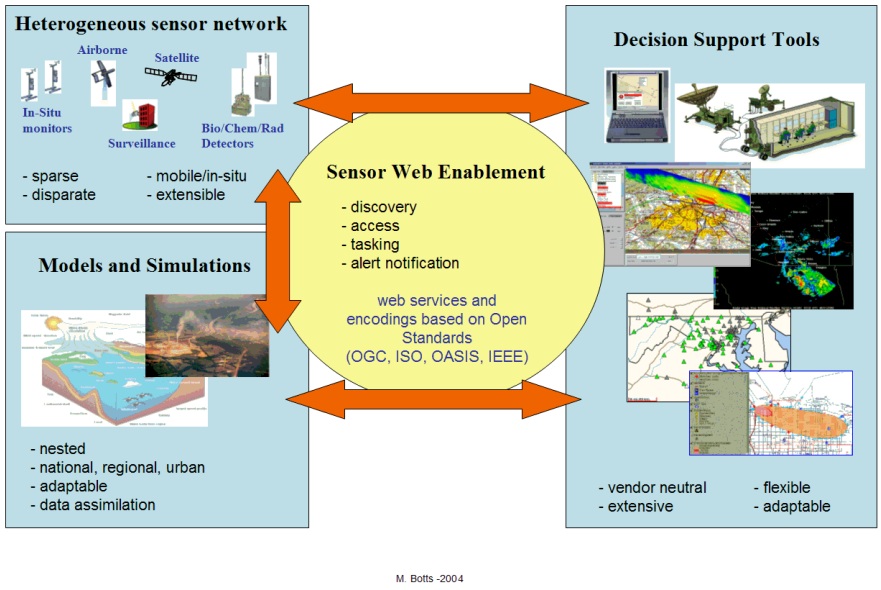


Abbildung 2. Die Rolle des Sensor Web Enablement Frameworks. (Quelle: [7])

einfache Datentypen verwendet werden, die sich falls nötig auch zu komplexen Datentypen zusammensetzen lassen.

Sensor Model Language (SensorML)[10] dient zur Beschreibung von Sensoren. Ursprünglich wurde SensorML als eine umfangreiche, XML-basierte Modellierungssprache für hochpräzise Sensoren wie zum Beispiel Satelliten entworfen. Durch die Eingliederung in SWE wurde es dann zu einer Beschreibungssprache für alle Arten von Sensoren erweitert. Die Praxistauglichkeit hat aber unter der anfänglichen Zielsetzung zu leiden, da die Beschreibung für die meisten Sensoren viel zu genau und umfangreich ist und ohne Kenntnis der Sensorhardware und Möglichkeiten zur exakten Positionsbestimmung kaum erstellbar ist. Schon die Beschreibung einer einfachen Wetterstation umfasst schnell tausende von Zeilen, der Auszug daraus in Listing 1 dient lediglich zur Festlegung des Standorts.

Anstatt einer detaillierten Beschreibung des internen Aufbaus verwendet SensorML ein funktionales Modell des Sensors, wobei sämtliche Komponenten eines Sensorsystems als Prozesse mit Ein- und Ausgabedaten, Parametern und Methoden betrachtet werden. Dies erlaubt eine Verknüpfung der Komponenten, um auch komplexe Anfragen bearbeiten zu können, die ein einzelner Sensor nicht erfüllen kann.

SensorML enthält neben der Prozessbeschreibung und den Positionsangaben auch Metadaten, die das (automatisierte) Auffinden von passenden Sensoren erleichtert, so kann zum Beispiel der vorgesehene Anwendungszweck eines Sensors angegeben werden.

```
...
<position name="stationPosition">
  <swe:Position localFrame="#STATION_FRAME"
               referenceFrame="urn:ogc:crs:EPSG:4329">
    <swe:location>
      <swe:Vector gml:id="STATION_LOCATION"
                  definition="urn:ogc:def:property:OGC:location">
        <swe:coordinate name="latitude">
          <swe:Quantity axisID="Y">
            <swe:uom code="deg"/>
            <swe:value>34.72450</swe:value>
          </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="longitude">
          <swe:Quantity axisID="X">
            <swe:uom code="deg"/>
            <swe:value>-86.94533</swe:value>
          </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="altitude">
          <swe:Quantity axisID="Z">
            <swe:uom code="m"/>
            <swe:value>20.1169</swe:value>
          </swe:Quantity>
        </swe:coordinate>
      </swe:Vector>
    </swe:location>
    <swe:orientation>
...
```

Listing 1. Auszug aus einem SensorML Dokument (Quelle: [10])

Transducer Markup Language (TML) besteht aus einer Menge von Modellen zur Beschreibung der Hardwareansprecheigenschaften von Messfühlern. Mit diesen Informationen können aus den rohen Messwerten zum Beispiel rauschfreie und latenzarme Sensordaten berechnet werden.

Des Weiteren definiert TML Methoden zum Transport von Sensordaten, wobei die Datenelemente in semantisch bedeutungsvollen XML Tags gespeichert werden. Diese Codierung eignet sich aufgrund des für XML typischen Umfangs allerdings nicht für die Übertragung in Echtzeit bei geringen Bandbreiten. Für diese Fälle wird ein minimalisiertes XML-Format verwendet, bei dem die Sensordaten in TML Clustern gruppiert werden.

Sensor Observation Service (SOS) ist die Vermittlungsschicht zwischen Benutzer und Sensordaten. Die standardisierten Webservice Schnittstellen ermöglichen das Anfragen, Filtern und Erhalten von archivierten oder in Echtzeit aufgenommenen Sensordaten, falls nötig übernimmt SOS hierfür auch die Steuerung der Sensoren. Zentraler Bestandteil von SOS sind Registrierungsdatenbanken mit Metadaten über die vorhandenen Sensoren und die von ihnen bereitgestellten Daten. Sie ermöglichen dem Client einen automatisierten Zugriff auf die von ihm benötigten Daten.

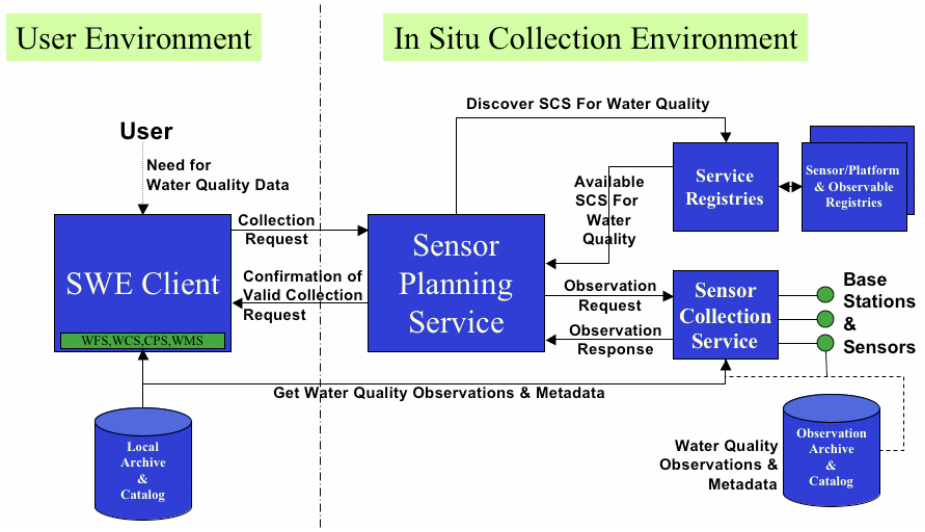


Abbildung 3. Verwendung von SPS zur Bestimmung der Wasserqualität. (Quelle: [11])

Sensor Planning Service (SPS) bietet eine Webservice Schnittstelle zur Planung von Anfragen an das Sensorsystem und dient als Vermittlungsschicht zwischen Benutzer und dem Sensorsystem. Der Client meldet, welche Art von Sensordaten benötigt werden, und SPS durchsucht automatisch seine Sensordatenbank und antwortet mit für den Client passenden und verfügbaren Sensoren. Mit diesen Informationen kann der Client dann mit Hilfe von SOS auf die Sensordaten selbst zugreifen. Abbildung 3 zeigt den beispielhaften Einsatz von SPS zur Bestimmung der Wasserqualität.

Sensor Alert Service (SAS) ermöglicht es Clients sich mithilfe standardisierter Webservice Schnittstellen für Alarmmeldungen anzumelden. Alarmmeldungen sind hierbei Mitteilungen über das Eintreten eines Ereignisses an einem bestimmten Objekt und enthalten einen Zeitstempel und eine Ortsangabe.

SAS dient allerdings nur als zentrale Verwaltungsstelle, bei der sich die Nachrichtenserver von Datenquellen registrieren. Sucht ein Client dort nach einer bestimmten Alarmmeldung so bekommt er als Antwort nur alle benötigten Informationen um sich mit dem passenden Nachrichtenserver zu verbinden. Dieser Verbindungsaufbau und der eigentliche Austausch der Alarmmeldungen sind nicht Bestandteil von SAS sondern benötigen weitere Protokolle wie zum Beispiel das OASIS Common Alert Protocol [12].

Web Notification Services (WNS) bietet Webservice Schnittstellen zur asynchronen Kommunikation mit SPS und SAS und anderen Diensten inner-

halb des SWE Frameworks.

SWE bietet somit ein umfangreiches Framework für Sensorbeschreibung, Sensor Discovery und für die Datenabfrage, allerdings nur in der Form von standardisierten Datenformaten und Schnittstellen, eine funktionsfähige Implementierung gibt es nicht. Es wird sich erst noch zeigen müssen, ob die allgemeinen und umfangreichen Standards tatsächlich für den Einsatz in der Praxis geeignet sind.

3.2 IrisNet

Das IrisNet (Internet-scale Resource-Intensive Sensor Network Services) Projekt [13] von Intel Research verfolgt das Ziel eines weltweiten Sensornetzes. Das Augenmerk liegt hierbei nicht nur auf den üblichen hoch spezialisierten Sensornetzwerkknoten sondern auch auf gewöhnlichen PCs, die sich dank weiter Verbreitung, hoher Rechenleistung und ständiger Internetverbindung gut als Sensoren eignen.

Abbildung 4 zeigt die zweistufige Architektur von IrisNet:

- Sensing Agents (SAs) bieten eine einheitliche Schnittstelle zum Zugriff auf Sensordaten, unabhängig vom Sensortyp.
- Organizing Agents (OAs) sind verteilte Datenbanken, die jeweils für einen einzelnen Dienst des Sensornetzwerks alle spezifischen Daten der SAs speichern.

Sensing Agents sammeln die rohen Sensordaten der angeschlossenen Sensoren. Prinzipiell sind alle Arten von Sensortypen verwendbar, der Fokus von IrisNet liegt aber auf Sensoren, die ein großes Datenvolumen verursachen und die für vielseitige Dienste verwendet werden können, zum Beispiel Webcams.

Um auf die Sensordaten zuzugreifen, kann ein Dienst auf dem SA eigene Programme ausführen. Diese sogenannten *Senselets* steuern die Sensoren, verarbeiten und filtern die rohen Sensordaten so wie sie für den Dienst benötigt werden und senden die Daten dann an den zuständigen OA. Durch die Verarbeitung direkt am Sensor kann zum einen die Netzwerklast signifikant reduziert werden, zum anderen wird hierdurch die Verarbeitung automatisch parallelisiert und die Rechenlast verteilt.

Damit ein SA gleichzeitig von mehreren Diensten verwendet werden kann ist jedes Senselet ein eigener Prozess. Diese Separation dient dem Schutz des SA und auch der Senselets untereinander, und ermöglicht außerdem eine Begrenzung der Ressourcen, die ein Senselet verbrauchen darf. Zur Reduzierung der Last auf dem SA können die Senselets aber über gemeinsamen Speicher automatisch bereits verarbeitete Daten austauschen, um zu verhindern, dass mehrere Senselets die gleiche rechenintensive Berechnung durchführen.

Zum Schutz der Privatsphäre wird mit Hilfe von digitalen Signaturen zwischen vertrauenswürdigen und nicht vertrauenswürdigen Senselets unterschieden. Nur vertrauenswürdige Senselets erhalten vollen Zugriff auf die rohen Sensordaten, für andere Senselets werden aus Sicht des Datenschutzes bedenkliche

Informationen automatisch ausgefiltert. So werden beispielsweise in Videodaten automatisch Gesichter erkannt und mit schwarzen Rechtecken überblendet.

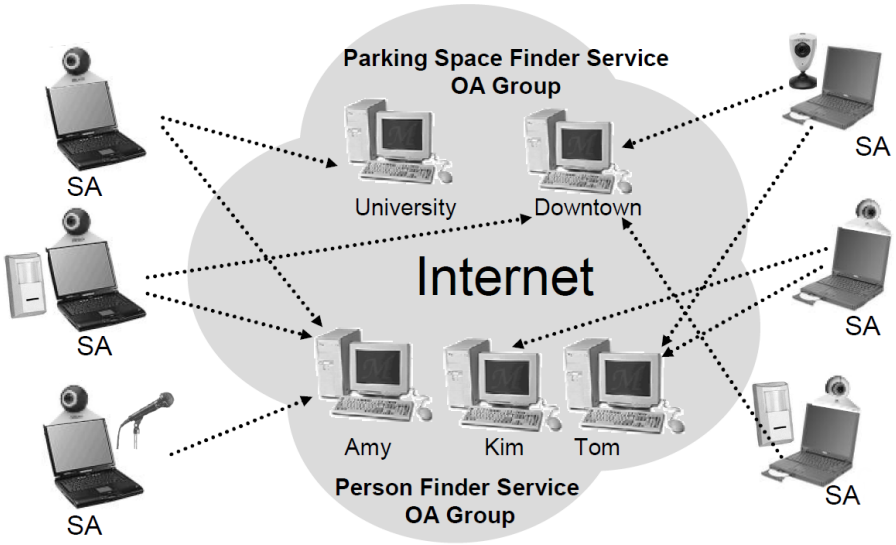


Abbildung 4. IrisNet Architektur. (Quelle: [13])

Organizing Agents dienen zum Verwalten und Speichern der von den SAs gesammelten Daten, wobei ein OA immer nur für einen bestimmten Dienst zuständig ist. Die Speicherung der Daten erfolgt in XML mit Hilfe von selbstbeschreibenden Tags, die anhand der geografischen Position hierarchisch organisiert werden. Damit sich IrisNet dynamisch an ändernde Lasten anpassen kann wird diese Hierarchie automatisch auf eine Menge von OAs verteilt. Durch diese Partitionierung werden die durchschnittliche Antwortzeit und die Netzwerkauslastung reduziert, während gleichzeitig versucht wird, die Last auf einem einzelnen OA nicht zu groß werden zu lassen.

```
/USRegion[@id='NE']/state[@id='PA']
/city[@id='Pittsburgh']/neighborhood[@id='Oakland']
/block[@id='block1' OR @id='block3']
/parkingSpace[available='yes']
```

Listing 2. XPATH Abfrage für freie Parkplätze (Quelle: [13])

Als Abfragesprache zum Zugriff auf die Daten der OAs eines Dienstes wird XPATH verwendet. Die Abfrage in Listing 2 dient zum Beispiel zur Bestimmung aller freien Parkplätze in Oakland in Block 1 oder Block 3. Anfragen werden solange entlang der Hierarchie weitergeleitet bis ein OA gefunden wird,

der die Anfrage beantworten kann. Damit mehrmalige Anfragen für ähnliche Daten beschleunigt werden, besitzt jeder OA einen Cache für bereits beantwortete Anfragen. Durch die Caches sind die Daten aber nicht immer aktuell, weshalb bei einer XPATH Anfrage angegeben werden kann, wie alt die Daten höchstens sein dürfen.

Die verteilte Speicherung der Daten wird auch zu Erhöhung der Robustheit und Ausfallsicherheit verwendet. Hierzu werden die gleichen Daten auf mehreren, örtlich möglichst weit voneinander entfernte OAs gespeichert.

Der Schwerpunkt von IrisNet liegt also auf der effizienten Datenverwaltung und Datenabfrage. Allerdings fehlt eine Beschreibung der Sensordaten und ein Verzeichnis der vorhandenen SAs und OAs, weshalb sich IrisNet hauptsächlich für Benutzer eignet, die nur ihre eigenen SAs und OAs bereitstellen und verwenden. Die Senselets sind ein interessanter Ansatz um die Heterogenität der Sensoren zu überbrücken und Netzwerklasten zu verringern, sie setzen aber entsprechend leistungsfähige Sensorplatformen voraus. Dass die Senselets vom Benutzer selbst erstellt werden müssen erschwert zudem das Hinzufügen von neuen Sensoren.

3.3 Service-Oriented Network proGramming of Sensors (SONGS)

SONGS [14] ist ein semantisches Sensorsystem, dessen Fokus auf einer möglichst einfachen Datenabfrage liegt.

Als Grundlage dient hierfür eine Ontologie, die hierarchisch beschreibt, welche Daten in einem Sensorsystem verfügbar sind, und zwar unabhängig davon, von welchem Sensor diese Daten bereitgestellt werden. Diese Abstraktion erleichtert es dem Benutzer sich auf die für ihn wichtigen Daten zu konzentrieren, ohne sich um unwichtige Details kümmern zu müssen. So spielt es für einen Verkehrsüberwachungsdienst beispielsweise keine Rolle ob die Geschwindigkeit mittels Lichtschranken oder Lasersensoren gemessen wird.

Zur Umwandlung der physikalischen Daten der Sensoren in die semantischen Objekte der Ontologie dienen sogenannte *semantic services*. Diese ereignisgesteuerten Softwarekomponenten bauen auf dem .NET Framework auf und nehmen als Eingabe die Daten eines Sensors oder eines anderen semantic services entgegen und erzeugen einen neuen semantischen Datenstrom. Zum Beispiel könnte ein Geschwindigkeitssensor als Eingabeereignis die Unterbrechung zweier Lichtschranken verwenden um als Ausgabe eine Geschwindigkeit zu liefern. Die Sensoren selbst werden als semantic services mit nur einem Ausgangsdatenstrom betrachtet. Als Beschreibung wird ein an logische Programmiersprachen erinnerndes Text-Format verwendet, in dem die Eingaben als Vorbedingungen und die Ausgaben als Nachbedingungen formuliert werden.

Um auf die Daten des Sensorsystems zuzugreifen werden semantische Abfragen verwendet, die nur beschreiben welche Daten der Ontologie benötigt werden, aber nicht wie diese beschafft werden sollen. Die Abfrage *photo(Car)* erzeugt zusammen mit der Servicebeschreibung aus Listing 3 zum Beispiel Bilder aller an vorhandenen Sensoren vorbeifahrenden Autos. Auf Gateway-Servern, die Informationen über alle vorhandenen semantic services besitzen, wird versucht die

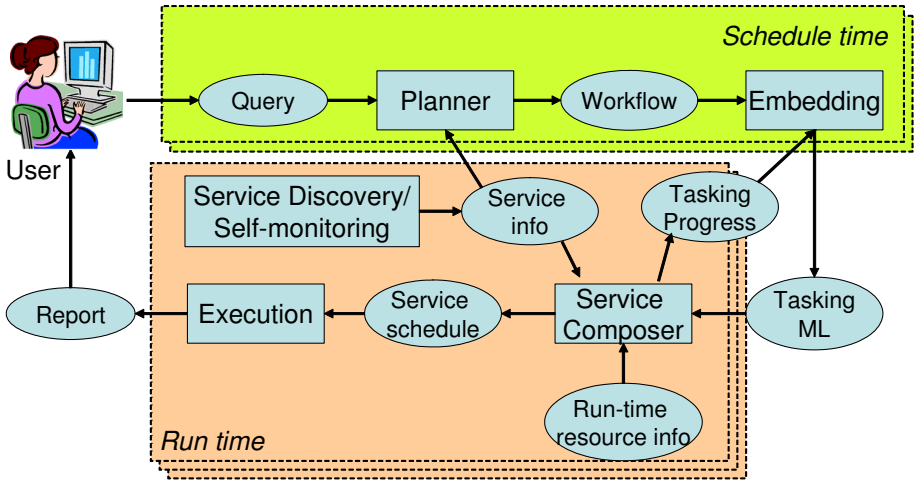


Abbildung 5. Datenabfrage in SONGS. (Quelle: [14])

Abfrage durch eine automatische logische Verknüpfung der semantic services zu erfüllen (siehe Abbildung 5). Ist diese Planung erfolgreich müssen die einzelnen Knoten des Ablaufgraphs jeweils einem semantic service zugeordnet werden. Hierbei werden Randbedingungen wie die Netzwerkauslastung, Latenz und die Verwendung von bereits vorhandenen Daten berücksichtigt um eine Überlastung des Systems zu verhindern. Bei der Ausführung der Abfrage liefert der letzte semantic service die gewünschten Daten an den Benutzer.

```
service(triggerablePhotoService(Y,Delta,Region),
  cstrnbl(Delta),
  needs(sensor(camera,Region),
        detected(Y,T2, )),
  creates(photo(X), detected(X,T,Region),
          triggered(X,Y), delay(T,T2,Delta)))
```

Listing 3. Servicebeschreibung in SONGS (Quelle: [14])

SONGS bietet also eine umfangreiche Sensorbeschreibung inklusive Ontologie und darauf aufbauend eine einfache Datenabfrage. Problematisch ist, dass eine gute Ontologie nur sehr schwer zu erstellen ist, denn wenn sie nicht allgemein genug ist geht der Vorteil gegenüber einem anwendungsspezifischen Sensorsystem verloren.

3.4 OntoSensor

Einen ähnlichen Ansatz wie SONGS verfolgt OntoSensor [15]. Auch hier wird eine hierarchische Ontologie verwendet, wobei die Elemente der Ontologie aber nicht die (verarbeiteten) Daten der Sensoren sondern die Sensoren selbst sind (siehe Abbildung 6). Somit eignet sich OntoSensor vor allem als gemeinsame

Sprache um die Eigenschaften, Funktionen, Attribute und Anforderungen verschiedenster Sensortypen in einer Wissensdatenbank semantisch einheitlich zu beschreiben, denn eine Verarbeitung der Sensordaten oder eine Verknüpfung mehrerer Sensoren, um semantisch höherwertige Daten zu erhalten, werden nicht unterstützt.

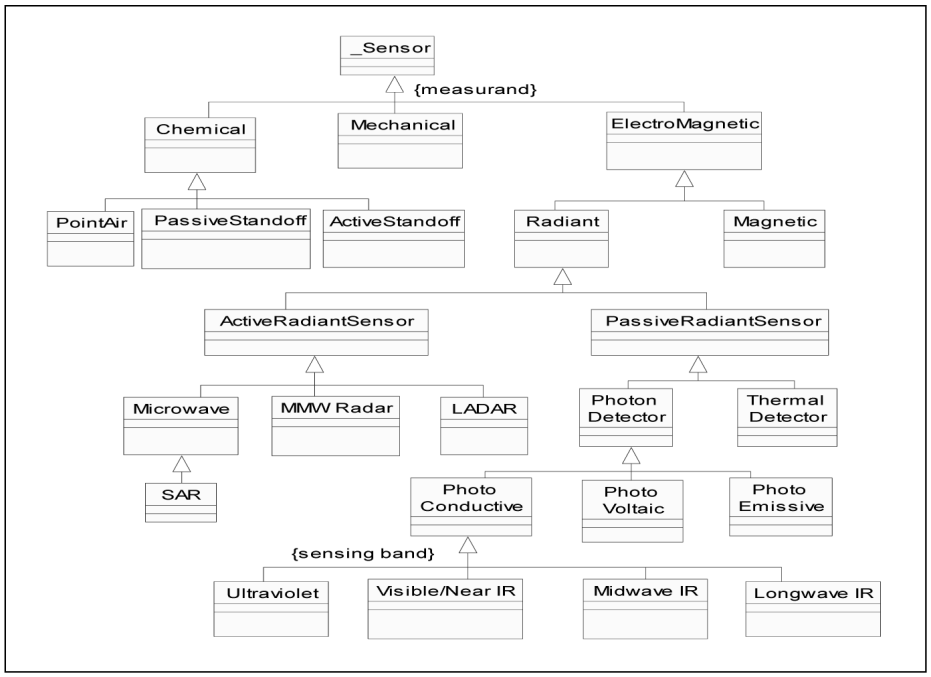


Abbildung 6. Auszug aus der Sensorhierarchie von OntoSensor. (Quelle: [15])

Als Grundlage für die Sensorberschreibung dienen SensorML und IEEE SUMO. Die XML Syntax allein von SensorML reicht aber nicht aus um semantisch hochwertige Sensorbeschreibungen zu erstellen, daher werden mit Hilfe von OWL semantische Informationen über die Sensoren und ihren Beziehungen untereinander hinzugefügt. Auch die Einordnungen der Sensoren innerhalb der Ontologie erfolgt auf diese Weise.

Für die Abfrage von Sensorinformationen aus dieser Wissensdatenbank wird, aufgrund der eingebauten logischen Folgerungsmöglichkeiten und der Unterstützung von Technologien des semantischen Webs, SWI-Prolog [16] verwendet. Das erlaubt flexible und allgemeine Anfragen, deren Antworten aus allen Informationen bestehen, die die Anforderungen der Anfrage erfüllen.

OntoSensor eignet sich also sehr gut dazu, Sensoren und ihre Beziehungen untereinander zu beschreiben und diese Informationen abzufragen. Um die Sensor-

daten selbst kümmert es sich aber nicht, weshalb es nur als Ergänzung zu einem anderen System, das die Sensordaten verwaltet, betrachtet werden kann.

3.5 SenseWeb

Das Ziel von SenseWeb [6] von Microsoft Research ist es, eine Infrastruktur aufzubauen, die es Anwendungen erlaubt auf die Daten von Sensoren zuzugreifen, die über das gesammte Internet verteilt sind. Die Sensordaten stammen hierbei von beliebigen Benutzern, die diese freiwillig zu Verfügung stellen. Dieser *community driven* Ansatz - ähnlich wie bei Projekten wie Wikipedia oder YouTube - ermöglicht es dank der gemeinsamen Nutzung der Daten durch effektive Koordinierung viele Sensoranwendungen einfacher und kostengünstiger zu realisieren als wie dies für eine allein stehende Anwendung möglich wäre. SenseWeb kümmert sich hierbei um die Auswahl der Sensoren für jede Anwendung und um die effiziente gemeinsame Nutzung eines Sensors von mehreren Anwendungen.

Die SenseWeb Architektur (siehe Abbildung 7) erlaubt einen einheitlichen Zugriff auf die heterogenen Daten und besteht aus *coordinator*; Sensoren, Sensorgateways und mobilen Proxies; *data transformers*; und Anwendungen.

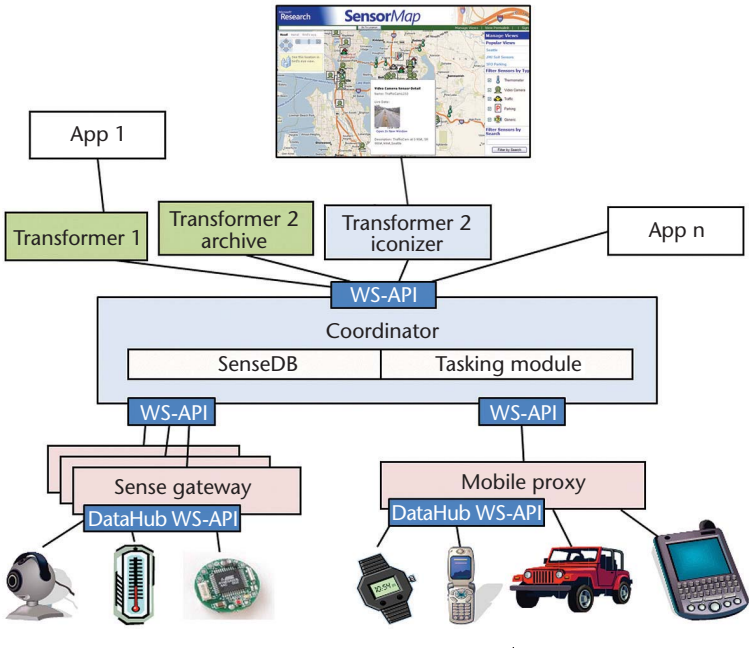


Abbildung 7. SenseWeb Systemarchitektur (Quelle: [6])

Coordinator
Der Coordinator, bestehend aus dem tasking Modul und der senseDB, bildet den zentralen Zugangspunkt in das System für alle Anwendungen und Sensordaten.

Das tasking Modul nimmt die Anwendungsanforderungen entgegen und versucht diese mit den zu verfügung stehenden Sensorresourcen zu erfüllen.

Die senseDB (streaming sensor database) dient zur effizienten Bearbeitung von sich überschneidenden Anwendungsanforderungen. Wenn mehrere Anwendungen Daten aus demselben Raumzeit-Fenster benötigen, versucht SenseDB die Last auf den zugehörigen Sensoren beziehungsweise Sensorgateways zu minimieren, indem es Anfragen zusammenfasst und einen Cache der kürzlich angeforderten Daten bereithält. Die Sensordaten können hier auch bereits aggregiert werden um speziellen Anfragen zu beantworten. So kann zum Beispiel der Mittelwert mehrerer Temperatursensoren berechnet werden wenn die Temperatur eines größeren Gebiets benötigt wird. Desweiteren speichert SenseDB auch einen Index mit Informationen über alle im System vorhandenen Resourcen. Realisiert wird die senseDB durch einen zentralisierten SQL-Server, das Caching und Aggregieren der Daten erfolgt mittels eines speziellen R-Baums.

Sensoren, Sensorgateways und mobile Proxies
Sensoren bilden die Grundlage von SenseWeb. Da es viele verschiedene Sensorplattformen mit unterschiedlichen Rechenleistungen, Energie- und Bandbreitenbedarf gibt, haben Sensoren häufig eine speziell auf sie zugeschnittene Schnittstelle. Um diese Heterogenität zu verstecken und die Sensoren in SenseWeb zu integrieren werden Sensorgateways verwendet. Diese kommunizieren einerseits über sensorspezifische Schnittstellen direkt mit dem Sensor, stellen andererseits aber eine gemeinsame WebService API bereit mit denen andere SenseWeb Komponenten auf die Sensordaten zugreifen können. Das Sensorgateway kann die Daten auch bereits filtern, und regeln wer auf die Daten zugreifen darf. So können autorisierte, und potenziell bezahlende, Nutzer beispielsweise die genauen Daten erhalten, während sich alle anderen Nutzer mit Näherungen begnügen müssen.

Die Beschreibung der Sensoren erfolgt mittels SDML (Sensor Description Markup Language), einem SensorML ähnlichen XML Format. Die Sensoren werden hierbei bezüglich ihren Darstellungs- und Aggregationsmöglichkeiten in Klassen mit gemeinsamen Eigenschaften eingeteilt, damit sie von der SenseDB und den data transformers automatisch bearbeitet werden können.

Wenn ein Benutzer seine Sensordaten zur Verfügung stellen will muss er also ein Sensorgateway bereitstellen, auf das SenseWeb zugreifen kann. Alternativ dazu gibt es auch den sogenannten DataHub, ein spezielles Sensorgateway, das von SenseWeb bereitgestellt wird und mit einer Vielzahl von Sensoren direkt kommunizieren kann. Über den DataHub werden auch Sensoren angebunden, die von SenseWeb mithilfe eines Webcrawlers gefunden wurden.

Mobile Proxies werden für mobile Sensoren verwendet. Sie ermöglichen einen transparenten Zugriff von Anwendungen auf diese Sensoren, unabhängig vom aktuellen Aufenthaltsort des Sensors.

Data transformers
Data transformers dienen zur semantischen Konvertierung von Daten. Ein transformer kann zum Beispiel aus dem Videobild einer Straße die Anzahl der vorbeifahrenden Autos pro Stunde extrahieren. Da die data transformers auch von Nutzern programmiert werden können, ist es möglich die Verarbeitungsmöglichkeiten von SenseWeb durch domänenspezifische Algorithmen zu erweitern. Die data transformers werden beim coordinator registriert und Anwendungen können diese je nach Bedarf verwenden.

Anwendungen
Anwendungen sind alle Nutzer von Sensordaten. Das können automatisierte Anwendungen im Enterprise Bereich sein, wie zum Beispiel das Erstellen von Kundenprofilen mithilfe von Videobildern, aber auch interaktive Anwendungen, bei denen manuell nach bestimmten Daten gesucht werden kann. Ein Beispiel für eine solche Anwendung ist SensorMap[4], das alle in SenseWeb verfügbaren Sensoren innerhalb eines Kartenausschnitts von Microsoft Live Maps automatisch je nach Zoomlevel aggregiert, diese anhand von data transformers in entsprechende Datenpunkte umwandeln lässt, und sie dann über der Karte einblendet.

SenseWeb ist das einzige der hier vorgestellten Systeme, das Lösungen für alle genannten Herausforderungen bietet, und es wird auch bereits erfolgreich in der Praxis für verschiedene Forschungsprojekte eingesetzt. Durch den modularen Aufbau ist es zudem auch auf zukünftige Änderungen gut vorbereitet, so könnte die zentrale senseDB zum Beispiel bei zu hoher Belastung relativ einfach durch eine verteilte Datenbank ersetzt werden.

4 Fazit

In Tabelle 1 sind die hier vorgestellten Sensorsysteme nochmals den Herausforderungen gegenübergestellt. Hierbei zeigt sich, dass sich mit Ausnahme von SenseWeb alle Systeme auf nur wenige Herausforderungen konzentrieren, um diese so gut wie möglich zu meistern. Die restlichen Herausforderungen werden dabei aber vernachlässigt, wodurch der Praxiseinsatz erschwert wird. Doch auch SenseWeb hat Schwächen, denn es bietet zwar Lösungen für die meisten Probleme, diese sind aber größtenteils nicht so umfangreich und mächtig wie die der anderen Systeme.

Problematisch ist auch, dass viele verschiedene, zueinander inkompatible Lösungen verwendet werden, um die Heterogenität von Sensoren zu überbrücken, wodurch die Heterogenität nur um eine Schicht nach oben verschoben wird. Besser wäre ein einheitlicher Standard, auf dem alle Systeme aufbauen können, und den dann auch die Hersteller von Sensoren unterstützen könnten.

Das beste Sensorsystem würde man wohl durch eine Kombination der Lösungen der einzelnen Systeme erhalten. So sticht zum Beispiel das Sensor Web Ena-

[4] http://atom.research.microsoft.com/sensormap/

Tabelle 1. Vergleich der Systeme anhand der Herausforderungen

	OGC SWE	IrisNet	SONGS	OntoSensor	SenseWeb
Sensorbeschreibung	SensorML und TML, selbstbeschreibendes XML	-	logische Beschreibung, in Ontologie eingebettet	Verknüpfung von SensorML, OWL und SUMO zu Ontologie	SDML
Sensor Discovery	Metadaten	geplant	-	-	Metadaten Webcrawler
Datenverwaltung	-	verteiltes XML	-	keine Sensordaten	zentraler SQL-Server
Datenabfrage	Sensor Observation und Planning Service	XPATH	Logische Erfüllung der Anfrage	Prolog	SQL
Visualisierung	Geoinformationssysteme	-	-	-	Datenpunkte in MS Live Maps

blement Framework durch seine offenen und genau spezifizierten Standards hervor, wodurch die Interoperabilität der einzelnen Komponenten erleichtert wird. Die beste Beschreibung der Sensoren selbst hingegen bietet OntoSensor mit seiner semantischen Wissensdatenbank. Für die Beschreibung der Sensordaten und die Datenabfrage bietet SONGS mit seiner Ontologie und logischen Verknüpfungen die beste Lösung, die effiziente Anfragen erlauben ohne das viel Detailwissen über das System erforderlich ist. Falls die Sensorplattformen über genügend Rechenleistung verfügen sind die senselets von IrisNet eine interessante Möglichkeit um das Netzwerk zu entlasten und die Sensordaten zu filtern. Die hierarchisch verteilte Datenbank bietet zudem eine hohe Ausfallsicherheit bei geringer Netzwerklatenz. Und SenseWeb verfügt über die besten Ansätze zur Integration des Sensorsystems in Web 2.0 Internetanwendungen wie zum Beispiel SensorMap, das eine Visualisierung der vorhandenen Sensoren bietet.

Für einige Herausforderungen existieren aber noch überhaupt keine überzeugenden Lösungen. Der Datenschutz wird von den Systemen, wenn überhaupt, nur ansatzweise unterstützt. Und auch die Sicherheit und Verlässlichkeit der Systeme ist gering, es gibt bisher so gut wie keine Möglichkeiten zu verhindern, dass das Sensorsystem - absichtlich oder durch Fehler - mit falschen Daten überflutet wird.

Literatur

1. World Wide Web Consortium: Extensible Markup Language (XML). http://www.w3.org/XML (2004)
2. Laerhoven, K.V., Berchtold, M., Gellersen, H.W.: DESCRIBING SENSOR DATA FOR PERVASIVE PROTOTYPING AND DEVELOPMENT. http://www.pervasive.ifi.lmu.de/adjunct-proceedings/poster/p023-027.pdf (2005)
3. Open Geospatial Consortium Inc: Sensor Model Language (SensorML) for In-situ and Remote Sensors. http://portal.opengeospatial.org/files/?artifact_id=7927 (2002)
4. World Wide Web Consortium: OWL Web Ontology Language. http://www.w3.org/TR/owl-features/ (2004)
5. Niles, I., Pease, A.: Origins of the IEEE Standard Upper Ontology. Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology, Seattle, 3742. (2001)
6. Kansal, A., Nath, S., Liu, J., Zhao, F.: SenseWeb: An Infrastructure for Shared Sensing. IEEE Multimedia. Vol. 14, No. 4 (2007)
7. Percivall, G., Reed, C.: OGC Sensor Web Enablement Standards. Sensors and Transducers Journal, Vol.71, Issue 9 (2006)
8. Open Geospatial Consortium Inc: Observation & Measurements Schema (O&M). http://portal.opengeospatial.org/files/index.php?artifact_id=14034 (2007)
9. World Wide Web Consortium: XML Schema. http://www.w3.org/TR/xmlschema-0/ (2004)
10. Open Geospatial Consortium Inc: OpenGIS Sensor Model Language (SensorML) Implementation Specification. http://xml.coverpages.org/OGC-06-010r6-TransducerMarkupLanguage-TML.pdf (2007)
11. Open Geospatial Consortium Inc: SWE Overview and High Level Architecture. http://portal.opengeospatial.org/files/?artifact_id=15540 (2006)
12. OASIS Emergency Management TC: OASIS Advances Common Alerting Protocol and Emergency Data Exchange Language. http://xml.coverpages.org/ni2005-05-19-a.html (2005)
13. Gibbons, P.B., Karp, B., Ke, Y., Nath, S., Seshan, S.: IrisNet: An Architecture for a World-Wide Sensor Web. IEEE Pervasive Computing, Volume 2, Number 4 (2003)
14. Liu, J., Zhao, F.: Towards Semantic Services for Sensor-Rich Information Systems. In: The 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks. (2005)
15. Russomanno, D.J., Kothari, C.R., Thomas, O.A.: Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In: The 2005 International Conference on Artificial Intelligence. (2005)
16. Wielemaker, J.: SWI-Prolog Reference Manual. http://swi-prolog.org/ (2000)

Probabilistische Zustandsschätzer in Ubiquitous Computing

Alexander Schütz aschuet@ira.uka.de

TecO Karlsruhe, Vincenz-Prießnitz-Str. 3 76131 Karlsruhe, Deutschland

Zusammenfassung Für Ubiquitous Computing ist es wichtig aus bestehenden Informationen ein Maximum an zusätzlichen Informationen zu gernerieren um daraus einen Kontext zu erstellen. Die hier beschriebenen Kalman Filter und Partikel Filter sind gute und gängige Methoden um dies zu erreichen, was an zwei Beispielen verdeutlicht wird.

1 Dynamische Zustandsschätzung für Ubiquitous Computing

Für Ubiquitous Computing ist es von enormer Bedeutung das Maximum an Information aus vorhandenen Messungen zu folgern, um daraus einen entsprechenden Kontext zu generieren. Dabei ist es wichtig viele unterschiedliche Messungen miteinander kombinieren zu können, wobei das beobachtete System meist dynamischen Veränderungen unterworfen ist. Probabilistische Ansätze bieten mit dem Kalman Filter und den Partikel Filtern eine gute Möglichkeit dies effektiv und effizient zu erreichen. Außerdem sind die genannten Verfahren relativ einfach zu implementieren und können die auftretenden Unsicherheiten schon von Natur aus modellieren.
In Abschnitt 2 und 3 werden zwei der weitverbreitetsten probabilistischen Methoden besprochen. Abschnitt 4 beinhaltet zwei Beispiele aus dem Bereich Ubiquitous Computing, bei denen diese Methoden zum Einsatz kommen. In Abschnitt 5 werden die wichtigsten Aspekte nochmals zusammenfasst.

2 Kalman-Filter

Der Kalman-Filter wurde erstmals 1960 als rekursive Lösung für das Problem der zeitdiskreten, linearen Filter von R.E. Kalman veröffentlicht. Er ist seit dem zu dem am häufigsten verwendeten Algorithmus zur dynamischen Zustandsschätzung avanciert [WB95]. Dank massiver Forschungen, gibt es heute eine Vielzahl an Erweiterungen zu diesem Filter, die seine Vorbedingungen abschwächen und ihn für eine größere Klasse von Problemen anwendbar machen.

2.1 Eigenschaften

Der Kalman Filter ist ein rekursiver Algorithmus zur dynamischen Zustandsschätzung. In seiner ursprünglichen Form verlangt er bestimmte Vorbedingungen an das

Vorabwissen und an das Prozess- und Sensormodell. Sind diese Bedingungen erfüllt so ist der Kalman Filter erwartungstreu, konsistent und beweisbar optimal, da er auf effiziente Weise den mittleren quadratischen Fehler minimiert[May79].

Linearität Das Prozessmodell das dem Kalman Filter zugrunde gelegt wird muß linear sein [May79], d.h. der Prozesszustand $x \in \Re^n$ verändert sich gemäß der folgenden Gleichung:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

Wobei die $n \times n$ Matrix A den Übergang von Zustand x_{k-1} nach x_k beschreibt. Die $n \times l$ Matrix B setzt die Steuerungsröße $u_{k-1} \in \Re^l$ mit dem Zustandswechsel in Beziehung. Der Prozessfehler w_{k-1} erweitert die Gleichung um eine Unsicherheit (Prozessrauschen)[WB95], welche durch die Modellierung eines physikalischen Sachverhaltes immer entsteht.
Analog zur Prozessgleichung, die aus dem Prozessmodell folgt, ist auch die Messgleichung definiert, die ihrerseits aus dem Sensormodell folgt. Sie setzt die Messwerte $z \in \Re^m$ durch die $m \times n$ Matrix H in Beziehung zu dem aktuellen Zustand und berücksichtigt dabei den Messfehler v_k. Die Gleichungen (1) und (2) beinhalten außerdem, dass es sich um einen Markov-Prozess 1. Ordnung handelt und somit der Folgezustand nur vom vorherigen Zustand abhängig ist und die Historie keine Rolle spielt[RN04].

$$z_k = Hx_k + v_k \tag{2}$$

[WB95]

Normalverteilt Eine zusätzliche Einschränkung zur Verwendung des Kalman Filters stellt die geforderte Gaußianität dar. Sowohl der Mess- und der Prozessfehler als auch der Initialzustand müssen normalverteilt sein [May79]. Außerdem müssen systematische Fehler ausgeschlossen sein, so dass der gemessene Wert im Mittel dem wahren Wert entspricht. Für die Fehlerverteilungsfunktionen $p(w)$ und $p(v)$ gilt dann:

$$p(w) \sim N(0, Q) \tag{3}$$
$$p(v) \sim N(0, R) \tag{4}$$

Q und R sind hierbei die Kovarianzmatrizen, der multivarianten Verteilungen [WB95].

Weißes Rauschen Der Prozess- und der Messfehler müssen weiterhin voneinander, von der Zeit und dem Zustand unabhängig sein. Die Amplitude des Rauschens muss also konstant sein. Somit ergäbe sich für das Rauschen eine unendlich große Energie. Da dies in der Praxis unmöglich ist, scheint diese Forderung utopisch. Allerdings reagiert jedes physikalische System nur auf

bestimmte Frequenzen (Frequenzband), in der Weise dass andere Frequenzen einen vernachlässigbar kleinen Einfluß auf das System haben. Es muss folglich nur gewährleistet sein, dass innerhalb dieses Frequenzbandes die Amplitude des Rauschens konstant ist .

Vorabwissen Um mit den Schätzungen beginnen zu können benötigt der Kalman Filter einige Informationen, die das System ausreichend beschreiben. Dies sind zum einen die Matrix R und die Matrix Q aus den Gleichungen (3) und (4). Sind beide konstant lässt sich R durch Testmessungen bestimmen während Q nur schlecht abgeschätzt werden kann, da der zu schätzende Zustand meist nicht direkt messbar ist. Beide können jedoch durch eine *Systemidentifikation* mittels eines weiteren Kalman Filters optimiert werden. Sind R und Q nicht konstant ist das darauf zurückzuführen, dass die Messgenauigkeit in der dynamischen Umgebungen stark variieren kann und auch die Prozessgenauigkeit eventuell vom Zustand abhängig ist. Je nach Zustand sollten beide neu abgeschätzt, beziehungsweise durch passendere R' und Q' ersetzt werden. Außerdem benötigt der Kalman Filter eine initiale Wahrscheinlichkeitsverteilung mit dem Startzustand x_0 als Erwartungswert und der Fehlerkovarianzmatrix P_0 [WB95].

2.2 Funktionsweise

Der Kalman Filter basiert auf dem Prinzip eines Prediktor-Korrektor Algorithmus zur Lösung numerischer Probleme.
Aus dem geschätzten Zustand $\hat{x}_k$ wird zunächst unter Verwendung des Prozessmodells ein Folgezustand $\hat{x}_k^-$ vorhergesagt und dessen Fehlerkovarianzmatrix P_k^- bestimmt.[1] [WB95]

$$\hat{x}_k^- = A\hat{x}_{k-1} + Buk - 1 \tag{5}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{6}$$

Unter den Bedingungen (3) und (4) ist die resultierende Verteilung ebenfalls eine Normalverteilung [Fra06] und es gilt:

$$p(x_k \mid x_{k-1}; z_{k-1}) \sim N(\hat{x}_k^-, P_k^-) \tag{7}$$

Dies wird als *a priori Schätzung* bezeichnet und entspricht einem kleinen Blick in die Zukunft. Damit ist die Prediktorphase, beziehungsweise die Aktuallisierung der Zeit komplett. In der darauf folgenden Korrekturphase, auch Messungsaktuallisierung genannt, wird die zum Zeitpunkt k durchgeführte Messung mit einbezogen um das Ergebnis der Vorhersage zu verbessern. Die resultierende *a posteriori Zustandsschätzung* erhält man aus der Linearkombination der *a priori Zustandsschätzung* und der gewichteten Differenz zwischen dem tatsächlich gemessenem Wert z_k und dem vorhergesagten Messwert $H\hat{x}_k^-$, der die Folgerung

[1] Bei den Zuständen $\hat{x}_k$ und $\hat{x}_k^-$ handelt es sich um geschätzte Zustände. Unter Zustand x_k ist der reale Zustand zu verstehen.

aus dem Sensormodell ist. Diese Differenz wird auch als *Innovation* bezeichnet.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{8}$$

Die $n \times m$ Gewichtungsmatrix K_k ist dabei der sogenannte *Kalman Gain*. Diese Matrix wird so gewählt, dass die *a posteriori Kovarianzmatrix* P_k minimiert wird. Eine mögliche Form von K_k wird durch die folgende Gleichung erreicht [WB95]:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{9}$$

An dieser Gleichung wird leicht ersichtlich, dass die Innovation bei genaueren Messungen mehr Gewicht erhält. Im Gegensatz dazu wird der *Kalman Gain* bei einem kleineren Prozessfehler geringer. Wenig verrauschten Messungen (kleines R) wird also mehr Glauben geschenkt und bei einem genauen Prozessmodell (kleines P_k^-) verlieren die Messungen an Bedeutung.
Zuletzt wird noch die *a posteriori Kovarianzmatrix* geschätzt, womit die Korrekturphase abgeschlossen ist.

$$P_k = (I - K_k H) P_k^- \tag{10}$$

Ebenso wie bei (7) ist auch die hier entstehende Verteilung eine Normalverteilung mit:

$$p(x_k \mid z_k) \sim N(\hat{x}_k, P_k) \tag{11}$$

Mit $\hat{x}_k$ und P_k als neue *a priori Parameter* kann nun erneut eine Vorhersage unternommen und der Algorithmus wiederholt werden. Der ganze Prozess ist nochmals in Abbildung 1 verdeutlicht.

2.3 Erweiterungen

Im Laufe der Zeit haben sich mehrere Ansätze entwickelt, die zum Ziel haben den Kalman Filter auf nicht lineare Systeme zu erweitern, da viele der praktischen Anwendungsgebiete in der einen oder anderen Weise nicht linear sind.

Extended Kalman Filter Der *Extended Kalman Filter* linearisiert hierzu die nichtlinearen Funktionen f (Prozessmodell) und h (Sensormodell) mit Hilfe des 1. Taylor-Polynoms um den zu schätzenden Zustand x_k. Dies funktioniert jedoch nur bei geglätteten, regelmäßigen Systemen [RN04]. Hinzu kommt, dass die Schätzungen vor allem bei stärker werdender Nichtlinearität, nicht mehr erwartungstreu sind und auch die Konsistenz nicht mehr garantiert ist. Genauere Näherungen durch Taylor-Polynome höherer Ordnung verbessern die Schätzungen im Allgemeinen nicht oder nur gering, so dass der dafür notwendige Aufwand ungerechtfertigt ist [Fra06].

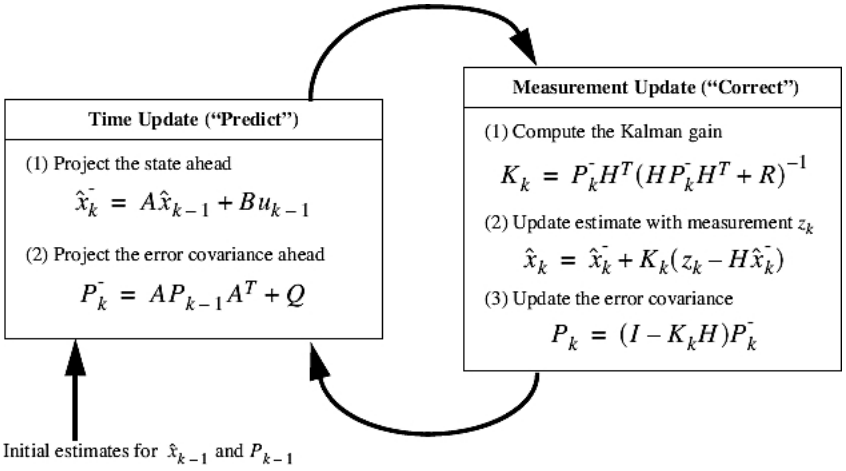


Abbildung 1. Schematische Darstellung des Prediktor-Korrektor Algorithmus inklusive der Updategleichungen

Unscented Kalman Filter Bessere Ergebnisse können meist mit dem von Julier und Uhlmann [JU04] vorgeschlagenen *Unscented Kalman Filter* erziehlt werden. Dabei wird die nichtlineare Funktion nicht linearisiert, sondern auf deterministische Art und Weise ein Menge gewichteter Sigmapunkte gewählt deren Erwartungswert und Kovarianzmatrix der geschätzten, realen Verteilung entsprechen. Die Sigmapunkte werden dann in die nichtlineare Funktion eingegeben und der neue *a priori Zustand* und dessen Kovarianzmatrix bestimmt. Heute wird meist die weiterentwickelte Form des UKF, der so genannte *skalierte unscented Kalman Filter* verwendet, da er in der Praxis das Problem einer nicht definiten Korrelationsmatrix vermeidet.[Fra06]

3 Partikel-Filter

Eine weitere rekursive Methode zur dynamischen Zustandsschätzung stellen Partikel Filter dar, die auch unter dem Namen Sequentiele Monte Carlo Methoden bekannt sind. Neben dem Kalman Filter sind Partikel Filter die bekanntesten porbabilistischen Zustandsschätzer, die auf Grund der gestiegenen Rechenleistung in den vergangenen Jahren immer mehr an Bedeutung gewonnen haben.

3.1 Eigenschaften

Sind die Prozessgleichung $x_k = f(x_{k-1}, v_{k-1})$ und die Messgleichung $y_k = h(x_k, w_k)$ zu starken Nichtliearitäten unterworfen liefern meist auch der *EKF* und der *UKF* keine ausreichende Näherung, insbesondere wenn das Prozess- und das Messrauschen nicht mehr normalverteilt sind. Partikel Filter machen

keine Annahmen über $p(v)$, $p(w)$ oder die entstehende Wahrscheinlichkeitsdichte $p(x_k \mid y_k)$ und können so beliebige, insbesondere Dichtefunktionen mit mehreren Maxima approximieren.

3.2 Bayesscher Filter

Das Vorbild von Partikel Filtern ist der optimale Bayessche Filter, den sie versuchen zu approximieren. Er basiert auch auf dem Prediktor-Korrektor Algorithmus. Im Gegnsatz zum Kalman Filter[2] werden das Prozessmodell und das Sensormodell nicht duch lineare Gleichungssysteme sondern durch bedingte Wahrscheinlichkeiten dargestellt, die aus dem Prozess- und Sensormodell folgen [AC02]. So bedingt das Prozessmodell die Übergangswahrscheinlichkeit $p(x_k \mid x_{k-1})$ und das Sensormodell die bedingte Wahrscheinlichkeit $p(z_k \mid x_k)$. Anhand der Formeln lässt sich erkennen, dass wieder die Markovannahme 1.Ordnung gilt. x_k ist also von der Historie $x_{0:k-2}$ unabhängig und nur vom direkten Vorgänger x_{k-1} abhängig [DA00]. Die Messungen sind ebenfalls unabhängig zueinander und hängen außerdem ebenfalls vom gegenwärtigen Zustand ab.
Das Ziel der Filterung besteht in der rekursiven Approximation der Dichte $p(x_k \mid y_{0:k})$ und der Vorhersage $p(x_k \mid y_{0:k-1})$. Diese lassen sich durch das Chapman-Kolmogorov Theorem (12) und durch Anwendung der Bayes-Regel mit der Markov Eigenschaft angeben als [Fra06] [AC02], [RN04]

$$p(x_k \mid y_{0:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid y_{0:k-1}) dx_{k-1} \tag{12}$$

$$p(x_k \mid y_{0:k}) = \frac{p(y_k \mid x_k) p(x_k \mid y_{0:k-1})}{p(y_k \mid y_{0:k-1})} \tag{13}$$

Der Ausdruck $p(y_k \mid y_{0:k-1})$ dient hierbei als Normalisierungsfaktor, der jedoch im Allgemeinen nicht analytisch bestimmt werden kann.[DA00] Gleichung (12) ist hierbei die Updateformel für die *Prediktionsphase* und analog Gleichung (13) für die *Korrekturphase*[AC02].

Sequential Importance-Sampling Grundlage fast aller Partikel Filter bildet der Monte Carlo Algorithmus des *sequentiellen importance-samplings* (*SIS*), bei dem die *a posteriori Dichte* $p(x_k \mid y_k)$ durch eine Menge von L gewichteten Stichproben, so genannten Partikeln $\{x^i_{0:k}, w^i_k\}^L_{i=1}$ approximiert wird [AC02]. Bei einer sehr großen Anzahl an Partikeln, entspricht *SIS* näherungsweise dem optimalen *Bayesschen Filter*.

$$p(x_k \mid y_{0:k}) \approx \sum_{i=1}^{L} w^i_k \delta(x_{0:k} - x^i_{0:k}) \tag{14}$$

$$wobei w^i_k \propto \frac{p(x^i_{0:k} \mid y_{0:k})}{q(x^i_{0:k} \mid y_{0:k})} und \sum_{i=1}^{L} w^i_k = 1. \tag{15}$$

[2] Genaugenommen handelt es sich beim Kalman Filter um eine spezialisierung des Bayesschen Filters (siehe [AC02]).

Die Gewichte w_k^i werden dabei durch die Methode des *Importance Samplings* bestimmt [Fra06], [AC02]. Da es meist nicht möglich ist Stichproben der Art $x_k^i \sim p(x_{0:k}^i \mid y_{0:k})$ zu generieren, werden beim *IS* die Stichproben so gezogen, dass $x_k^i \sim q(x_{0:k}^i \mid y_{0:k})$ gilt. Die Funktion $q(x_{0:k}^i \mid y_{0:k})$ wird *Importance Dichte* genannt und ist entscheidend für eine gute Approximation. Für die *Importance Dichte* gilt weiterhin

$$q(x_{0:k} \mid y_{0:k}) \stackrel{!}{=} q(x_k \mid x_{0:k-1}, y_{0:k}) q(x_{0:k-1} \mid y_{0:k-1}) \tag{16}$$

da sie unabhängig von der zukünftigen Messung sein soll [Fra06]. Durch die Umformungen

$$p(x_{0:k} \mid y_{0:k}) = \frac{p(y_k \mid x_{0:k}, y_{0:k-1}) p(x_{0:k} \mid y_{0:k-1})}{p(y_k \mid y_{0:k-1})} \tag{17}$$

$$= \frac{p(y_k \mid x_{0:k}, y_{0:k-1}) p(x_k \mid x_{0:k-1}, y_{0:k-1}) p(x_{0:k-1} \mid y_{0:k-1})}{p(y_k \mid y_{0:k-1})} \tag{18}$$

$$\propto p(y_k \mid x_{0:k}, y_{0:k-1}) p(x_k \mid x_{0:k-1}, y_{0:k-1}) p(x_{0:k-1} \mid y_{0:k-1}) \tag{19}$$

und die zuvor geforderte Faktorisierung von $q(x_{0:k} \mid y_{0:k})$ ergibt sich die Updategleichung für die *Importance Gewichte* unter Berücksichtigung der Markovannahme zu

$$w_k^i \propto w_{k-1}^i \frac{p(y_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, y_k)}. \tag{20}$$

Der *SIS* Algorithmus bietet hervorragende Möglichkeiten zur Parallelisierung, er hat jedoch das Problem, dass die Varianz der *Importance Gewichte* monoton steigend ist und somit schon nach wenigen Iterationsschritten häufig alle bis auf wenige Stichproben ein vernachlässigbar kleines Gewicht haben. Dieses Problem wird *Sampledegeneration* genannt und ist dafür verantwortlich, dass keine gute Approximation mehr möglich ist. Ein Maß für die Degeneration erhält man durch

$$N_eff = \frac{1}{\sum_{i=1}^{L} (w_k^i)^2} \tag{21}$$

welches eine Näherung der effektiven Stichprobenanzahl N_eff [AC02] ist.

Resampling Neben der Möglichkeit, durch geeignetere *Importance Dichten* der Stichprobendegenerierung zu begegnen gibt es die einfache und weit verbreitete Variante des *Resampling (mit Ersetzen)*, welches beim Unterschreiten einer unteren Schranke durch N_eff durchgeführt wird. Beim *Resampling* wird eine neue Menge an Partikeln $\{\widetilde{x}_k^{*i}, \widetilde{w}_k^{*i}\}_{*i=1}^{L}$ mit $\widetilde{w}_k^{*i} = \frac{1}{L}$ aus den Partikeln $\{x_k^i, w_k^i\}_{i=1}^{L}$ generiert, die dadurch ersetzt werden. Dabei werden Stichproben mit großem w_k^i vervielfältigt, im Gegensatz zu Stichproben mit niedrigem Gewicht, die verworfen werden. Des weiteren wird gefordert

$$P(\widetilde{x}_k^j = x_k^i) = P(i(j) = i) \stackrel{!}{=} w_k^i \tag{22}$$

Eine einfache Möglichkeit dies zu erreichen ist das *systematische Resampling*, bei dem eine gleichverteilte Zufallsvariable $u(1)$ auf dem Intervall $[0, \frac{0,1}{L}]$ erzeugt wird. Die Zuordnung von neuen zu alten Partikeln $i(j)$, beziehungsweise die Neuindexierung erfolgt dann wie folgt

$$i(j) \stackrel{!}{=} \arg\min_i u(j) \leq c(i) \tag{23}$$

wobei $u(j) = u(1) + \frac{j-1}{L}$ und $c(i) = \sum_{\nu=1}^{i} w_k^\nu$. Dieses Verfahren kann in $O(L)$ implementiert werden und ist somit auch effizient. Durch das Resampling entsteht unter Umständen jedoch das neue Problem der *Sample-Verarmung*, da bei geringem Prozessrauschen die Stichproben an den Stellen der Vervielfältigung ähnlich bleiben, wodurch der Zustandsraum nur noch unzureichend abgedeckt ist. Weiterhin erschwert das *Resampling* eine Parallelisierung, da die Partikel kombiniert werden müssen [AC02].

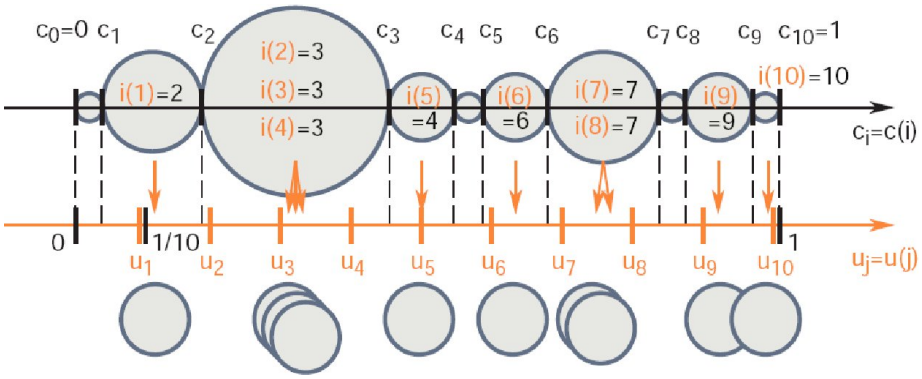


Abbildung 2. Die Indexierung bei *systematischem Resampling*

Regularisierung Der *Sample-Verarmung* kann zum einen durch ein absichtliches Überhöhen des Prozessrauschens entgegen gesteuert werden. Dies führt in der Regel zum gewünschten Erfolg, hat jedoch meist eine schlechtere Konvergenz des Filters zur Folge. Zum anderen kann die Methode der *Regularisierung* verwendet werden.[AC02] Beim *Regularisieren* wird eine Verrauschung der neuen Stichproben beim *Resampling* vorgenommen, beim dem nicht mehr von der diskreten Approximation (14) ausgegangen wird, sondern von einer so genannten *Kernel-Approximation* der Form

$$p(x_k \mid y_{0:k}) \approx \sum_{i=1}^{L} w_k^i K_B(x_{0:k} - x_{0:k}^i) \tag{24}$$

$$K_b(x) = \frac{1}{B^{\dim x}} K(\frac{x}{B}) \tag{25}$$

Der *Epanechnikov-Kernel* stellt die *optimale Kernel-Funktion* dar, da er den *asymptotic mean integrated squared error (AMISE)* minimiert [Kir06]. Weiter verbreitet ist jedocht die Verwendung eines *Gauß-Kernels*, da er die Berechnungen vereinfacht [DDG01]. Der *Gauß-Kernel* hat dabei die folgende Form mit der dazugeörigen *Kovarianzmatrix* $\widetilde{P}_k$ und *Bandbreite* [Fra06]

$$K(x) = N(0, \widetilde{P}_k) \tag{26}$$

$$B = [\frac{4}{L(2 + \dim x)}]^{\frac{1}{4+\dim x}} \tag{27}$$

Die verrauschten, neuen Stichproben $\overset{*}{x}{}_k^j$ ergeben sich dann aus

$$\overset{*}{x}{}_k^j = \widetilde{x}_k^j + B\widetilde{P}_k^{\frac{1}{2}} n_k^j \tag{28}$$

wobei $n_k^j \sim N(0, 1)$ gilt. Die verwendete Kovarianzmatrix wird vor dem *Resampling* bestimmt, um die bereits vorhandene Genauigkeit mit berücksichtigen zu können.
Da eine *Regularisierung* nur bei einem *Resampling* vorgenommen wird, wird ein zu starkes Verrauschen wie bei der Methode des überhöhten Prozessrauschens verhindert [Fra06].

SIR und RPF Wie bereits erwähnt spielt die Wahl der *Importance Dichte* eine sehr große Rolle im Hinblick auf die Genauigkeit des Filters. Viele der bekanntesten Partikel Filter unterscheiden sich fast ausschließlich in der Wahl der *Importance Dichte* und lassen sich leicht aus den bisherigen Methoden herleiten. Der *Sampling Importance Resampling* Partikelfilter ist einer der gebräuchlichsten Schätzer, da er nicht nur gute Ergebnisse für eine Vielzahl von Problemklassen liefert, sondern auch, wie gezeigt werden wird mit wenig arithmetischen Aufwand verbunden ist. Außerdem ist seine Funktionsweise intuitiv leicht zugänglich [Fra06]. Bei *SIR* wird die *Importance Dichte* gleich der Übergangswahrscheinlichkeit aus dem Prozessmodell gewählt.

$$q(x_k^i \mid x_{k-1}^i, y_k) \overset{1}{=} p(x_k \mid x_{k-1}) \tag{29}$$

Außerdem wird ein *Resampling* zu jedem Zeitschritt durchgeführt [3], wodurch sich die Gewichte zu $w_k^{i-1} = \frac{1}{L} \forall i$ ergeben. Setzt man dies nun in die Gewichtsupdategleichung (17) ein, so ergibt sich

$$w_k^i \propto p(y_k \mid x_k^i) \tag{30}$$

wobei die Gewichte wieder normalisiert werden, so dass $\sum_i w_k^i = 1$. Die Auswahl der Stichproben der Form $x_k^i \sim p(y_k \mid x_{k-1}^i)$ gestaltet sich ebenfalls als einfach, wenn Stichproben der Art $v_{k-1}^i \sim p(v_{k-1})$ aus dem Prozessrauschen erzeugt

[3] Dies kann auch mit einer *Regularisierung* verbunden werden.

werden können. Die neuen Stichproben egeben sich dann aus den vorherigen Partikeln, die durch das verrauschte Prozessmodel propagiert wurden.

$$x_k^i = f(x_{k-1}^i, v_{k-1}^i) \tag{31}$$

SIR kann durch das häufige *Resampling* jedoch sehr schnell zu einer *sample Verarmung*, besonders bei Prozessen mit geringem Prozessrauschen führen. Dies kann wie bereits erwähnt durch *Regularisierung* vermieden werden. Der entstehende *regularisierte Partikel Filter (RPF)* weißt im Allgemeinen eine bessere Abdeckung des Schätzbereiches auf, als *SIR* [Fra06], was jedoch nicht zwingend zu einer besseren Schätzung führt [AC02].

4 Anwendungsbeispiele in Ubiquitous Computing

Für Ubiquitoous Computing spielt vor allem die Möglichkeit der Kontexterkennung, beziehungsweise Abschätzung durch Partikel Filter eine große Rolle. So ist es unerlässlich die Possition eines Kunden zu bestimmen, wenn diesem ortsspeziefische Services erbracht werden sollen. Die folgenden beiden Beispiele werden sich auf dieses zentrale Thema der Ortsbestimmung und des Trackings beschränken. Weitere Anwendungen von Partikel Filtern zur Abschätzung relevanter Kontexte sind jedoch ohne weiteres möglich.

4.1 Kontexterkennung in Videokonferenzen

Ein Beispiel der Kontextabschätzung ist gegeben durch [ZD02]. Dieses System wird in intelligenten Videokonferrenzumgebungen, bei Verhaltensstudien von Tieren und zur Besprechungsprotokollierung verwendet.
Um mehrere Objekte (in diesem Fall Menschen) zu lokalisiern und ihre Bewegung im Raum zu verfolgen, werden mehrere unterschiedliche Sensordaten aus Audio und Videomesssystemen genutzt und in einen *SIR* Partikel Filter kombiniert. Ein weiteres Ziel des entwickelten Systems ist das Ermitteln der Zuordnung eines gesprochenen Textes zu einem Sprecher. Dazu gehört insbesondere die Erkennung der Redeordnung. Außerdem werden personalisierte Protokolle der Besprechung erstellt und mit dem Partikel Filter auch die intrinistischen Systemparameter, also die Anordnung der Sensoren zueinander, bestimmt.

Realisierung Der Zustandsvektor (hier s_t), des Systems wird möglichst klein gehalten und ergibt sich zu $[x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2]$ mit dem Ortsvektor $[x, y, z]$, dem Beschleunigungsvektor $[\dot{x}, \dot{y}, \dot{z}]$ und den Rotationswinkeln und Rotationsgeschwindigkeiten $[\phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2]$ zweier rotierender Mirkrofonarrays. Die Prozessgleichungen sind wie folgt definiert

$$x(t + \delta t) = x(t) + \dot{x}\delta t \tag{32}$$

$$\dot{x}(t + \delta t) = \dot{x}(t) + F\delta t \tag{33}$$

wobei F ein Prozessrauschen darstellt, welches normalverteilt mit dem Erwartungswert Null und der Standardabweichung σ ist, die je nach geschätzter Beschleunigungsfähigkeit und Manövrierfähigkeit gewählt wird. Analog ergeben sich die Prozessgleichungen für y, z und ϕ.
Bei der Videolokalisierung wird zuerst eine Gesichtserkennung durchgeführt, indem Farbsegmente gesucht werden die einem Oval bestimmter minimaler Größe entsprechen und dann verglichen wird, ob Gesichtsmerkmale vorhanden sind. Die Projektion der realen Koordinaten auf die Bildkoordinaten und umgekehrt erfolgt mittels einer linearen Transformation, deren Parameter durch eine Kalibrierung erhalten werden können. Danach startet das Tracking bei dem, ausgehend von dem Punkt, der aus der Prozessgleichung im Bildkoordinatensystem vorhergesagt wird, ein erneutes Matching durchgeführt wird. Auf diese Weise können mehrere Objekte gleichzeitig verfolgt werden, ohne dass das gesamte Bild gematcht werden muss.

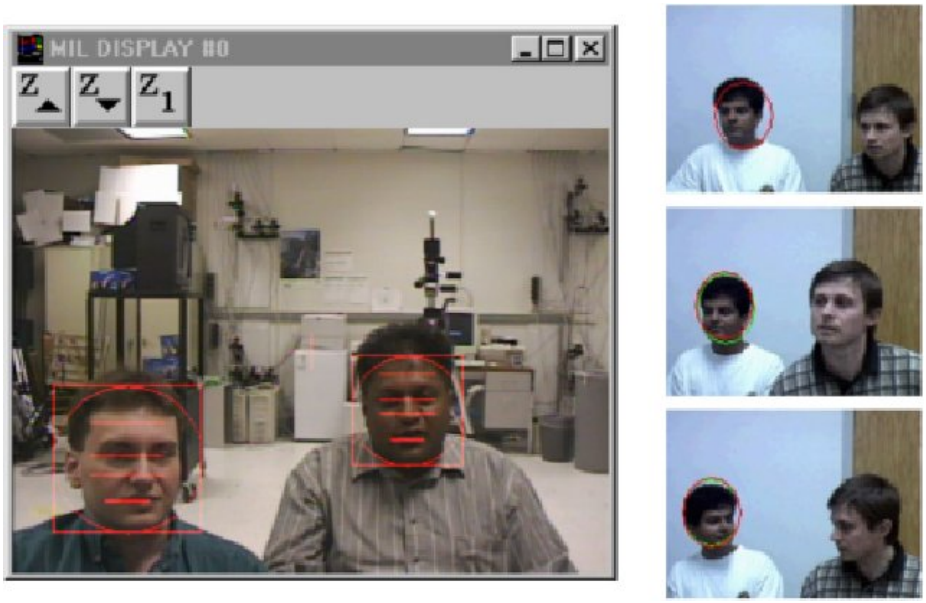


Abbildung 3. Screenshot der Gesichtserkennung und drei Bilder einer Trackingsequenz

Die Audiolokalisation wird Anhand der Zeitdifferenz zwischen den Ankünften der Signale durchgeführt. Mit der dadurch gewonnenen Positionsschätzung können durch Verwendung eines Beanforming-Algorithmus, Störgeräusche eliminiert und die Signalqualität verbessert werden, so dass die Audiodaten auch durch ein Spracherkennungssystem ausgewertet werden können.

Die aus den sensormodellen entstehenden bedingten Wahrscheinlichkeiten $p(Y_{vid}(k) \mid S_k) \sim N(0, \sigma_{vid})$ und $p(Y_{aud}(k) \mid S_k) \sim N(0, \sigma_{aud})$ werden zusammengefasst zu

$$p(Y_k \mid S_k) = p(Y_{vid}(k) \mid S_k) \sim N(0, \sigma_{vid}) p(Y_{aud}(k) \mid S_k) \sim N(0, \sigma_{aud}) \quad (34)$$

um das Tracking durchzuführen.

Ergebnisse Im Ergebnis zeigt sich, dass dieses System nicht nur in der Lage ist die Stimmen den anwesenden Personen zuzuordnen, sondern auch durch die Kombination der Daten eine bessere Genauigkeit und Robustheit gegenüber Ausfällen beim Tracking aufweist, als dies das Video-, beziehungsweise das Audiotracking erreichen. Es ist ebenfalls möglich die Positionen der Sensoren zueinander während dem Tracking zu bestimmen und auch partielle und vollständige Verdeckung sind gut handhabbar. Anzumerken ist, dass die Möglichkeiten der Kontextbestimmung in diesem Beispiel noch nicht ausgereizt ist. Es ist auch denkbar mit Partikel Filtern die jeweilige Stimmungslage der einzelnen an den Konferenzen teilnehmenden Personen und ihre Verhalten gegenüber den anderen abzuschätzen, indem Lautstärke und Bewegungsintensität als Messgrößen verknüpft werden.

4.2 Weiträumiges Tracking

Ein weiteres Beispiel des Einsatzes von Partikel Filtern in Ubiquitous Computing ist die Verwendung von bereits weitverbreiteten Technologien zur Lokalisation und zum Tracking von Menschen in weitläufigen Gebieten.
Da die Kosten einer abdeckenden Sensorenausbringung in einer Großstadt zu hoch sind, werden bereits vorhandene Lokalisationstechniken, wie GPS benutzt. Leider hat es sich gezeigt, dass diese an den Orten, an denen sich Menschen am häufigsten aufhalten nur eine geringe Abdeckung[4] besitzen. So ist GPS in Gebäudekomplexen oft nicht verfügbar und eine genaue Lokalisation damit nicht zufriedenstellend. Auch Techniken wie GSM, WLAN und Bluetooth lösen solche Probleme nur teilweise. So hat GSM zwar eine hohe Abdeckung, aber es bietet auch nur eine geringe Genauigkeit. Diese steigt zwar bei der Verwendung von WLAN, allerdings ist eine gute Abdeckung nur in Großstädten und nur tagsüber zu erwarten [LCC+05]. Außerdem sind die Positionen der meisten Access Points nicht bekannt, da diese in privatem Gebrauch sind.
PlaceLab[LCC+05] bietet einen guten Lösungsansatz für diese Probleme. Hierbei werden die vorhandenen Technologien gemeinsam für eine Lokalisation benutzt und dadurch sowohl eine nahezu komplette Abdeckung als auch eine gute Schätzgenauigkeit erzielt.
Durch die Verwendung der GSM und der WLAN Technologie konnte sogar in ländlichen Gebieten sowohl eine ganzheitliche Abdeckung gewährleistet werden, als auch eine mittlere Genauigkeit von etwas mehr als 30m, die mit steigender

[4] Mit Abdeckung ist in diesem Beispiel nicht die Geographische Abdeckung, sondern die zeitliche Abdeckung des durchschnittlichen Lebensraums gemeint.

Einwohnerzahl pro km^2 steigt (siehe Tabelle 1). Die Positionen der unbekannten Access Points werden dabei aus War-drive-logs generiert.
Zur Lokalisation werden ein einfacher Tracker, der die Schnittflächen der empfangenen Signale berechnet und ein Partikel Filter benutzt. Dieser verwendet die abnehmende Signalstärke und die Häufung von Signalverlusten bei steigender Distanz zum Signalemitter als Grundlage für sein Sensormodell. Zwar ist der einfache Tracker resourcensparender, doch mit dem Partikel Filter können zum einen genauere Ergebnisse geliefert werden und zum anderen bietet er eine größere Fülle an zusätzlichen Informationen, wie der Geschwindigkeit. Dadurch können unter Umständen weitere Genauigkeitsgewinne erzielt werden. Der einfache Tracker wird deshalb nur auf weniger leistungsfähigen Geräten, wie Mobiltelefonen genutzt.

	802.11		GSM		802.11 + GSM	
	accuracy	coverage	accuracy	coverage	accuracy	coverage
Downtown Seattle (Urban)	20.5 m	100.0%	107.2 m	100.0%	21.8 m	100.0%
Ravenna (Residential)	13.5 m	90.6%	161.4 m	100.0%	13.4 m	100.0%
Kirkland (Suburban)	22.6 m	42.0%	216.2 m	99.7%	31.3 m	100.0%

Tabelle 1. Die ermittelten Genauigkeiten und Abdeckungen nach [LCC+05]

5 Fazit

Wie an den Beispielen gesehen werden konnte eignen sich Partikel Filter gut zur Informationsgewinnung und somit auch für die Verwendung in Ubiquitären Systemen. Da in den vergangenen Jahren die Rechenleistung stark gestiegen ist, erfreuen sie sich einer zunehmenden Beliebtheit. Forschungen haben diverse Erweiterungen präsentiert, die teilweise den Aufand reduzieren und gute Werte liefern. Partikel Filter bleiben jedoch im Gegensatz zum Kalman Filter nur eine Approximation der optimalen Zustandsschätzung, so dass die Verwendung des letzteren immer der Verwendung von Partikel Filtern vorzuziehen ist, wenn die entsprechenden Vorbedingungen erfüllt sind.

Literatur

[AC02] Maskell S. Gordon N. Arulampalam, M.S. and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE transactions on signal processing*, 50:174–188, 2002.

[DA00] Godsill S. Doucet, A. and Ch. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

[DDG01] Arnaud Doucet, Nando Defreitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, June 2001.

[Fra06] Dietrich Fraenken. Dynamische Zustandsschätzung. Skript zu einer Vorlesung in der Fakultät für Elektrotechnik, Informatik und Mathematik der UniversitÂ¨at Paderborn, 2006.

[JU04] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92:401–422, 2004.

[Kir06] Prof. Thomas Kirste. KSWS Smart Environments - Nichtlineares, nichtgausssches Tracking. 2006. Vorlesungsfolien zu einer Vorlesung in der Fakultät für Informatik der UniversitÂ¨at Rostock.

[LCC+05] Anthony Lamarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. *Pervasive Computing*, 3468:116–133, 2005.

[May79] P.S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, August 1979.

[RN04] St. Russel and P. Norvig. *Künstliche Intelligenz - Ein moderner Ansatz*, volume 2. Pearson Studium, August 2004.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. 1995.

[ZD02] Duraiswami R. Zotkin, D.N. and L.S. Davis. Joint audio-visual tracking using particle filters. *EURASIP Journal on Applied Signal Processing*, 11:1154–1164, 2002.

Unscharf - Fuzzy Systeme

Florian Kriebel

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer: Martin Berchtold

1 Einleitung

Seit dem Aufkommen der unscharfen Mengentheorie wird ihre Brauchbarkeit und Korrektheit im Vergleich zur Wahrscheinlichkeitstheorie in Bezug auf Anwendungen diskutiert. Im Hinblick darauf tauchen folgende Fragen auf: Welche Unterschiede und Gemeinsamkeiten bestehen zwischen beiden Theorien? Macht eine die andere überflüssig? Wann ist welche vorzuziehen? Die Meinungen zu diesen Fragen gehen weit auseinander - von der völligen Ablehnung der unscharfen Mengentheorie bis hin zu großer Begeisterung.
Im Folgenden soll zuerst ein kurzer Überblick über die unscharfe Mengentheorie gegeben und deren Eigenschaften dargestellt werden. Zudem werden Unterschiede zur Wahrscheinlichkeitstheorie dargelegt. Im Anschluss daran werden einige Argumente präsentiert, die für und gegen diese neue Theorie sprechen bzw. in welchen Fällen sie genutzt werden kann und in welchen die Wahrscheinlichkeitstheorie vorzuziehen ist. Diese Punkte sollen daraufhin an realen Beispielen im Bereich der ubiquitären Informationssysteme vertieft und die Entscheidung für die jeweilige Wahl der Methode verdeutlicht werden. Zum Abschluss sollen die wichtigsten Erkenntnisse zusammengefasst werden.

2 Unschärfe

2.1 Unscharfe Mengen

Gemäß Klir [1] sind unscharfe Mengen - ähnlich wie scharfe - über eine charakteristische Funktion definiert.
Bei unscharfen Mengen wird die Zugehörigkeit zu einer Menge nicht durch eine ja/nein-Entscheidung (0 oder 1) festgelegt, sondern sie kann jeden Wert zwischen 0 und 1 annehmen. Die Funktion zur unscharfen Menge A über dem Universum U sieht somit wie folgt aus:

$$\mu_A : U \rightarrow [0, 1]$$

Folglich beschreibt μ_A den Grad der Zugehörigkeit eines Elements zu einer unscharfen Menge. Diese Zugehörigkeitsfunktionen bieten weiterhin die Möglichkeit Mengen darzustellen, welche durch unscharfe linguistische Begriffe beschrieben werden.
Wie Klir [1] erläutert, können die grundlegenden Operationen scharfer Mengen

wie Komplement, Schnitt und Vereinigung auf unscharfe Mengen erweitert werden. Es wird jedoch darauf hingewiesen, dass diese nicht mehr eindeutig sind, sondern eine Vielzahl an Möglichkeiten existieren. Als Beispiel nennt Klir [1] die folgenden Standardoperationen auf unscharfen Mengen (dabei sind A und B unscharfe Mengen und μ_A bzw. μ_B deren Zugehörigkeitsfunktionen):

$$\overline{A} : \mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

$$A \cup B : \mu_{A \cup B}(x) = max\left\{\mu_A(x), \mu_B(x)\right\}$$

$$A \cap B : \mu_{A \cap B}(x) = min\left\{\mu_A(x), \mu_B(x)\right\}$$

Diese Definitionen führen allerdings dazu, dass einige Gesetze verletzt werden, welche bei scharfen Mengen gelten. Dies ist zum einen das Gesetz der ausgeschlossenen Mitte $A \cup \overline{A} = U$ und das Gesetz des Widerspruchs $A \cap \overline{A} = \emptyset$, so dass es möglich ist, dass ein Element teilweise zu einer Menge und zu ihrem Komplement gehört.
Jedoch gibt es nach Hanebeck [3] auch einige Begriffe, die bei scharfen Mengen nicht existieren. Als Beispiele seien hier zum einen der α-Schnitt genannt, welcher für eine unscharfe Menge A und $\alpha \in [0, 1]$ folgendermaßen definiert ist: $A_\alpha = \{x \in U : \mu_A(x) \geq \alpha\}$. Hier ist anzumerken, dass es sich bei A_α um eine scharfe Menge handelt. Weiterhin entspricht der 0-Schnitt dem Universum. Zudem wird $supp(A) = \{x \in U : \mu_A(x) > 0\}$ als Trägermenge („Support“) der unscharfen Menge bezeichnet. Übderdies weist Hanebeck [3] darauf hin, dass unscharfe Mengen durch ihre α-Schnitte eindeutig definiert sind. Klir [1] nennt zudem noch den Begriff der „normalen unscharfen Menge“, wobei eine unscharfe Menge A als solche bezeichnet wird, wenn für die Höhe $h(A)$=1 gilt. Dabei ist $h(A)$ der größte Wert der Zugehörigkeitsfunktion μ_A. Ein Beispiel für eine Zugehörigkeitsfunktion und die weiteren genannten Begriffe ist in Abbildung 1 zu sehen.

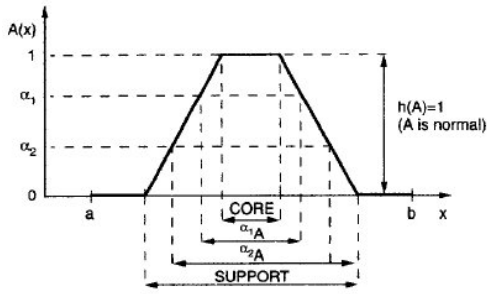


Abbildung 1. Beispiel für eine Zugehörigkeitsfunktion, Trägermenge, α-Schnitt und Höhe [1]

2.2 Unscharfe Logik

Um schließlich zu einer unscharfen Logik zu gelangen, wird die Zugehörigkeitsfunktion nach Klir [1] als „Wahrheitsgrad“ der Aussage „x ist der unscharfen Menge A zugehörig“ aufgefasst. Hierbei korrespondieren die logischen Operationen Negation, Konjunktion und Disjunktion mit den Operationen Komplement, Vereinigung und Schnitt auf unscharfen Mengen, wobei auch hier viele Alternativen möglich sind. Yen [4] weist allerdings darauf hin, dass die Wahl von Konjunktion und Disjunktion nicht unabhängig geschehen kann, um weiterhin die De Morganschen Gesetze zu erfüllen. Die möglichen unscharfen Konjunktionsoperatoren werden schließlich als T-Normen bezeichnet, die unscharfen Disjunktionsoperatoren als S-Normen bzw. T-Conormen. Beispielhaft seien hier die Eigenschaften von T-Normen gemäß Yen [4] genannt, wobei $t(x, y)$ ein T-Norm Operator ist $(w, x, y, z \in [0, 1])$:

1. t(0,0)=0, t(x,1)=t(1,x)=x
2. Monotonie: $t(x, y) \leq t(z, w)$, wenn $x \leq z$ und $y \leq w$
3. Kommutativität: $t(x, y) = t(y, x)$
4. Assoziativität: $t(x, t(y, z)) = t(t(x, y), z)$

Darüber hinaus macht Hanebeck [3] darauf aufmerksam, dass auch mehrere Möglichkeiten exisitieren eine unscharfe Implikation zu definieren. Dies ist ein Unterschied zum scharfen Fall, wo Konjunktion, Disjunktion und Negation ausreichen, um alle anderen Operatoren herzuleiten. Ein Beispiel ist die Darstellung der Implikation als „nicht-oder“ $(x \rightarrow y \equiv \overline{x} \vee y)$, bei entsprechend vorgegebenen Normen für Negation und Disjunktion. Die Implikation bildet die Grundlage für den Modus Ponens, ein logisches Schlussmuster.

2.3 Unscharfes Schließen

Um trotz Unschärfe letztendlich auch Schlüsse zu ermöglichen, existieren ebenfalls mehrere Möglichkeiten, wie etwa das possibilistische oder linguistische Schließen. Für letzteres kann gemäß Hanebeck [3] eine unscharfe Erweiterung des Modus Ponens definiert werden, wobei folgende Begriffe eingeführt werden, die [3] entnommen sind:

- **linguistischer Term:** Eine unscharfe Menge A heißt linguistischer Term auf U gdw. die Zugehörigkeitsfunktion μ_A ein auf U definiertes Konzept der natürlichen Sprache repräsentiert.
- **linguistisches Prädikat:** Sei A ein linguistischer Term auf U und $x \in U$, dann heißt der Ausdruck „$x\ is\ A$“ linguistisches Prädikat auf U.
- **linguistische Variable:** Sei $x \in U$ und sei zusätzlich ein linguistisches Prädikat „$x\ is\ A$“ gegeben. Dann heißt x linguistische Variable auf U. Eine linguistische Variable repräsentiert damit eine unscharfe Beobachtung.
- **linguistische Regel:** Seien „$x\ is\ A$“ und „$y\ is\ B$“ linguistische Prädikate auf U bzw. V. Dann heißt der Ausdruck „$IF\ x\ is\ A\ THEN\ y\ is\ B$“ linguistische Regel.

Der auf unscharfe Mengen verallgemeinerte Modus Ponens lautet dann wie folgt:

$$\begin{array}{ll} IF\ x\ is\ A\ \ THEN\ y\ is\ B & \text{(Regel)} \\ x\ is\ A' & \text{(Fakt)} \\ \hline y\ is\ B' & \text{(Schluss)} \end{array}$$

wobei A und B linguistische Terme auf U bzw. V sind und $x \in U$ und $y \in V$ Variablen.
Diese unscharfen IF-THEN-Regeln, welche Bedingungen bezüglich einer linguistischen Variable zu einem Schluss führen, sind nach Yen [4] eine der am häufigsten genutzten Anwendungen. Dies führt er auf die Tatsache zurück, dass es möglich ist Schlüsse zu ziehen, selbst wenn die Bedingungen nur teilweise erfüllt sind.

2.4 Diskussionen über unscharfe Logik

Im Folgenden sollen nun einige Argumente in der Diskussion über unscharfe Logik präsentiert werden und in diesem Zusammenhang auch Unterschiede zwischen unscharfer Logik und Wahrscheinlichkeitstheorie dargestellt werden.
Hanebeck [3] und Almond [7] nennen als grundlegenden Unterschied die verschiedene Betrachtung von Ungewissheit. Während die Wahrscheinlichkeitstheorie Mittel bietet um *Unsicherheit* zu beschreiben, ist unscharfe Logik eine Möglichkeit, die es erlaubt, mit *Unschärfe* umzugehen. Doch wie Bezdek [5] erläutert, wurde bereits die Existenz von nichtstatistischer Ungewissheit in Zweifel gezogen. Er selbst ist der Ansicht, dass diese beispielsweise aufgrund linguistischer Ungenauigkeiten existiert. Jedoch argumentiert die Gegenseite, dass die mangelhafte Verwendung natürlicher Sprache nicht als Beweis für die genannten Ungenauigkeiten gesehen werden kann.
Laviolette [6] stellt schließlich vier Hauptunterschiede der Theorien dar. Zum einen führt er die bereits genannte Tatsache an, dass in der unscharfe Mengentheorie das Gesetz der ausgeschlossenen Mitte nicht gilt, was in der klassischen Mengentheorie jedoch der Fall ist. Zum anderen nennt er das Argument der Befürworter der Theorie, Phänomene der realen Welt seien vage, was der von der Wahrscheinlichkeitstheorie eingeführten Präzision widerspreche. Weiterhin wird dargelegt, die unscharfe Mengentheorie könne genutzt werden um intuitives menschliches Handeln nachzubilden und könne somit menschliche Entscheidungsfindung beschreiben. Diesem wird allerdings entgegnet, der Mensch habe einen proabilistischen Instinkt, welcher es aufgrund von akuraten Bewertungen zukünftiger Alternativen ermöglicht habe zu überleben. Allerdings existiert ebenso die Ansicht, Merkmale einer Spezies würden sich nicht stabilisieren, sondern sich im Hinblick auf Veränderungen der Umwelt ständig anpassen.
Als letzten Punkt nennt Laviolette [6] die Argumentation einiger Verfechter der unscharfen Mengentheorie, dass die Axiome der Wahrscheinlichkeitstheorie ein Spezialfall derer der unscharfen Mengentheorie seien und letztere sei aufgrund dieser Tatsache ersterer überlegen.

Almond [7] stellt einige Probleme beider Theorien dar. Er nennt die Vielzahl an Bedeutungen einer Zugehörigkeitsfunktion und darüber hinaus die Schwierigkeit die vielen möglichen Aspekte (Wahl der T-Norm, Aussehen der Zugehörigkeitsfunktion) bei unscharfer Logik sinnvoll zu wählen. Jedoch räumt er ein, dass dieses Argument auch auf die Wahrscheinlichkeitstheorie zutrifft, wenn die notwendige Erfahrung nicht vorhanden ist. Weiterhin führt Almond [7] an, dass der unscharfen Logik eine Entscheidungstheorie fehlt und keine „Lernfähigkeit“, wie etwa bei Bayesnetzen, vorhanden ist. Als Schwächen der Wahrscheinlichkeitstheorie sieht er die Notwendigkeit, mit Unabhängigkeitsannahmen vorsichtig umzugehen und das Beharren auf präzisen Formulierungen.
Zadeh [2] hingegen ist das Auffassung, dass die Theorien nicht konkurrierend angelegt, sondern eher unterschiedliche Anwendungsbereiche gegeben sind. Dennoch nennt er einige Einschränkungen der Wahrscheinlichkeitstheorie. So bietet diese keine Möglichkeiten mit unscharfen Ereignissen umzugehen, weshalb keine Schlüsse aus Voraussetzungen gezogen werden können, welche durch die „reale Welt“ gegeben sind. Ebenso existieren keine Mittel, um mit unscharfen Quantifizierern (z.B. viele, die meisten, ...) und Wahrscheinlichkeiten (z.B. wahrscheinlich, nicht sehr wahrscheinlich) umzugehen. Zudem wird die Analyse von Problemen, welche mit unscharfen Ausdrücken beschrieben sind, erschwert. Zadeh [2] trennt schließlich die Anwendungsgebiete: die Wahrscheinlichkeitstheorie ist weiterhin in Gebieten anwendbar, welche keine nennenswerten menschlichen Schlussfolgerungen oder Emotionen beinhalten (z.B. Quantenmechanik, Kommunikationssysteme), wohingegen der Verwendungsbereich der unscharfen Mengentheorie genau dort angesiedelt ist, wo dies eine große Rolle spielt (z.B. Sprach-, Mustererkennung).

3 Anwendungen

Zur Verdeutlichung und Veranschaulichung der bisher genannten Punkte werden nun einige Anwendungen aus dem Bereich der ubiquitären Informationssysteme vorgestellt. Zuerst wird der Einsatz von unscharfer Logik mit Hilfe zweier Beispiele präsentiert, danach die Anwendung der Wahrscheinlichkeitstheorie an zwei Beispielen. Am Ende steht schließlich eine kurze Darstellung wie man beide Ansätze gemeinsam verwenden kann.

3.1 Unscharfe Logik

Anwendung 1: Zustandsdarstellung eines ubiquitären Robotersystems
Im Zentrum der Arbeit von Guarino und Saffiotti [9] steht die Fragestellung wie ein komfortables und natürliches Interface zwischen einem Benutzer und einem komplexen System gestaltet werden kann. Obwohl heutige ubiquitäre Systeme aus vielen heterogenen, miteinander verbundenen Geräten bestehen, sollen diese vom Nutzer dennoch als ein System gesehen werden. Insofern sollen sowohl die Heterogenität als auch die Komplexität des Systems vor dem Nutzer verborgen werden.

Die Funktionalität des Systems soll weiterhin durch die Kooperation vieler einfacher Komponenten statt der Nutzung komplexer Roboter erbracht werden. Jedes Gerät wird dabei als „PEIS“ (Physically Embedded Intelligent System) dargestellt. Dabei handelt es sich um eine Menge verbundener Softwarekomponenten, welche Anschlüsse zu Sensoren und Aktuatoren haben können. Zudem wird ein gemeinsames Kommunikationsmodell verwendet, das den Austausch von Informationen zwischen PEIS erlaubt. Auf Basis eines gemeinsamen Kooperationsmodells ist es einem PEIS weiterhin möglich, Funktionalitäten anderer zu verwenden.
Ein Beispiel für solche Komponenten könnte ein Staubsauger sein, welcher nur über Reinigungsfunktionalität verfügt, aber mithilfe eines Trackingsystems seine Position im Haushalt bestimmen könnte. Das bedeutet, man schließt beide PEIS zu einer „PEIS-Ecology“ zusammen, um das Genannte zu ermöglichen. Zudem könnte eine Pflanze ebenfalls mit einem PEIS bestückt werden, der Informationen über diese liefert (Feuchtigkeit, statisch/bewegbar), um dem Staubsauger mitzuteilen ob diese verschiebbar ist oder nicht.
Um dem Problem der Heterogenität zu begegnen, besteht nun die Frage, wie man Statusinformationen verschiedener Geräte des ubiquitären Systems - also unterschiedlichen PEIS - gemeinsam handhabbar macht. Beispielsweise könnte der Energievorrat des Staubsaugers zur Neige gehen oder die Pflanze Wasser benötigen. Hier werden jedoch unterschiedliche Maße erkennbar. Eine Lösung für dieses Problem ist die Einführung einer abstrakten Sicht „Zufriedenheit“, deren Grad mithilfe unscharfer Werte zwischen 0 (total unzufrieden) und 1 (total zufrieden) repräsentiert wird. Zur Veranschaulichung kann erneut der Staubsauger dienen, wo die PEIS-Komponente „Batteriemanager“ den Zustand der Batterie überwacht. Der Grad der Zufriedenheit wird durch das unscharfe Prädikat „Batterie“ (siehe Abbildung 2) dargestellt, wobei τ_{empty} und τ_{full} für eine leere bzw. vollständig geladene Batterie stehen.

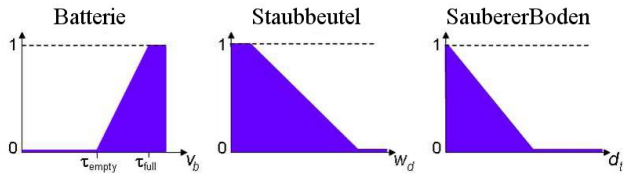


Abbildung 2. unscharfe Zufriedenheitsprädikate des Staubsaugers [9]

Diese Nutzung unscharfer Prädikate erlaubt eine gemeinsame Repräsentation des Zustands aller PEIS und deren Kombination mithilfe von unscharfer Logik. So kann auch ein Zufriedenheitsstatus für die gesamte PEIS-Ecology bestimmt werden. Um diesen für Menschen erkennbar zu machen, wird die Darstellung

eines Gesichts verwendet, dessen Ausdruck je nach Zustand zwischen traurig (0) und glücklich (1) liegt.
Um dies an einem Beispiel zu verdeutlichen, werden noch die Prädikate „Staubbeutel“ (Bestimmung des Füllstands des Staubbeutels durch internen Sensor) und „SaubererBoden“ (Bestimmung des in den letzten Minuten aufgesammelten Schmutzes durch Schmutzsensor) eingeführt. Weiterhin könnte die Pflanze Prädikate „Feuchtigkeit“ und „Temperatur“ bereitstellen. Die Zufriedenheit kann also durch die Formel

$$\Phi = ((Batterie \wedge Staubbeutel) \vee SaubererBoden) \wedge (Feuchtigkeit \wedge Temperatur)$$

dargestellt werden. Der Zufriedenheitsstatus zum Zeitpunkt t wird dann durch $\Phi(t)$ repräsentiert, welcher durch Regeln der unscharfen Logik bestimmt wird:

$$\Phi(t) = ((Batterie(t) \otimes Staubbeutel(t)) \oplus SaubererBoden(t)) \otimes (Feuchtigkeit(t) \otimes Temperatur(t))$$

wobei $\otimes$ jede T-Norm (verschiedene Möglichkeiten der unscharfen Verallgemeinerung der logischen Konjunktion) und $\oplus$ die korrespondierende T-Conorm (verschiedene Möglichkeiten der unscharfen Verallgemeinerung der logischen Disjunktion) sein kann.
Ein Beispiel zeigt Abbildung 3, wo für $\otimes/\oplus$ min/max verwendet wird.

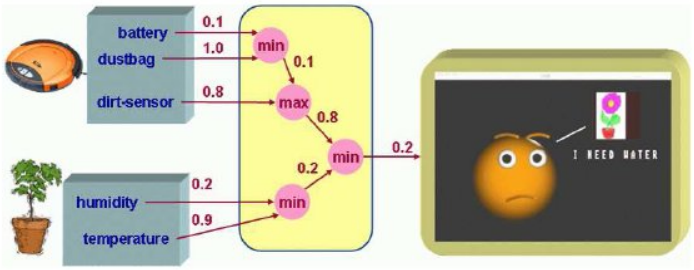


Abbildung 3. Beispiel für Berechnung des Gesamtzustands des Systems und Interface zum Benutzer [9]

Mit dieser Methode zur Bestimmung des Zufriedenheitsstatus werden folgende Anforderungen erfüllt: Die Berechnung kann...

– ...komplexer sein als lediglich den Durchschnitt oder das Minimum zu berechnen, was beispielsweise durch eine freie Wahl der T-Norm/T-Conorm ermöglicht wird.

– ...modular und dezentralisiert erfolgen, was durch ein Aufteilen der Berechnung von $\Phi(t)$ erreicht werden kann. So könnte jede PEIS lokal die Zufriedenheit ihrer PEIS-Komponenten kombinieren, was im oben genannten Beispiel folgendermaßen aussehen könnte:

$$\Phi_1(t) = (Batterie(t) \otimes Staubbeutel(t)) \oplus SauberesBoden(t)$$

$$\Phi_2(t) = Feuchtigkeit(t) \otimes Temperatur(t)$$

$$\Phi(t) = \Phi_1(t) \otimes \Phi_2(t)$$

Diese Möglichkeit der freien Verteilung besteht aufgrund dessen, dass T-Normen/T-Conormen assoziativ und kommutativ sind.

– ...die Zurückverfolgung des Grundes für einen geringen Zufriedenheitsstatus erlauben. Für die Verwendung von min/max kann der Gesamtzustand in disjunktive Normalform ($\Phi = \bigvee_{i=1}^{n} (\bigwedge_{j=1}^{k_i} l_j)$) gebracht werden. So kann zuerst der Konjunktionsterm i und danach das Literal l_j mit dem kleinsten Wahrheitswert bestimmt werden.

Die Verwendung der unscharfen Logik bietet bei dieser Anwendung einige Vorteile: Zum einen erlaubt die Wahl der T-Norm/T-Conorm eine gewisse Flexibilität und deren Eigenschaften ermöglichen eine „Rückverfolgung“ des Grunds für geringe Zufriedenheit. Dieses Konzept kommt weiterhin dem menschlichen Schlussfolgern/Handeln recht nahe und auch das Interface zum menschlichen Benutzer, welches die Zufriedenheit über unterschiedliche Ausdrücke repräsentiert, lässt eine Art „Emotion“ erkennen.

Anwendung 2: Anpassung von Anwendungen mobiler Terminals mit Nutzung unscharfer Kontextinformationen

Mäntyjärvi und Seppänen [10] präsentieren einen Ansatz zur Nutzung von unscharfer Logik, welcher Kontextinformationen nutzt, um Applikationen zu steuern und anzupassen. Ein Hauptgrund für die Nutzung unscharfer Logik liegt hierbei in der Eigenschaft, dass diese im Gegensatz zur boolschen Logik stetige Kontrollsignale ermöglicht. Weiterhin können Kontextinformationen in einem einheitlichen Format dargestellt werden.
Um eine Kontextrepräsentation durchzuführen, werden Sensoren verwendet. Nach einer Vorverarbeitung der Sensorsignale werden Merkmale extrahiert, die am geeignetsten erscheinen, den Kontext in der realen Welt wiederzugeben. Schließlich findet eine Quantisierung statt, um eine aussagekräftige Darstellung zu erhalten. Diese wird mithilfe unscharfer Mengen durchgeführt, die gegenüber einer scharfen Repräsentation den Vorteil bieten, keine reinen wahr/falsch-Aussagen zu treffen. Ein Beispiel wäre die Einordnung der Umgebungslautstärke, wie sie in Abbildung 4 zu sehen ist.

Die Vorgehensweise zur Erstellung von Kontextinformationen kombiniert quantifizierte Merkmale in einen Kontextatomvektor $x(n) = [x_1, x_2, ..., x_k]^T$ zu jeder Quantisierungszeit n, wobei die Werte einzelner Atome zwischen 0 und 1 liegen.

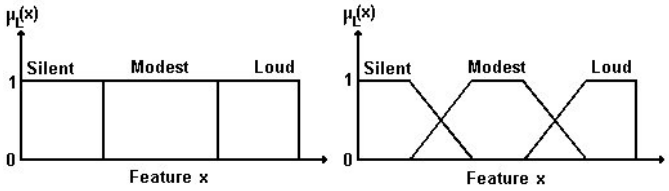


Abbildung 4. Gegenüberstellung: scharfe und unscharfe Repräsentation der Lautstärke [10]

Zur Steuerung der Applikationen werden „Fuzzy Logic Controller“ verwendet. Ein solcher besteht nach Laviolette [6] aus vier Operatoren: dem Fuzzifier, einer Regelbasis, einer Entscheidungslogik und einem Defuzzifier. Ersterer wandelt eine scharfe Messung in Zugehörigkeiten zu bestimmten unscharfen Mengen um, woraufhin die Regelbasis entscheidet, welche Aktion stattfindet. Treten dabei Konflikte auf, werden diese durch die Entscheidungslogik aufgelöst und schließlich im letzten Schritt in eine scharfe Aktion umgewandelt. Jeder Fuzzy Logic Controller benutzt als Eingabe eine vordefinierte Menge von Kontexten und eine Regelbasis, um mit vielen unscharfen Kontexten umzugehen. Die vordefinierte Menge enthält hierbei die wichtigsten Kontexte aus Sicht der Anwendung.
Als Beispiel stellen Mäntyjärvi und Seppänen [10] ein Experiment vor, in dem Inhalt und Informationsdarstellung kontextabhängig dargestellt werden. Dies sind Lautstärke von Betriebsgeräuschen, Textgröße, Displaybeleuchtung und die Kompression von Inhalten. Ein Laptop verarbeitet die Daten verschiedener Sensoren mithilfe einer Kontexterkennungssoftware, die unscharfe Mengen nutzt, um drei Kontexttypen zu beschreiben: „Nutzeraktivität“, „Lautstärke der Umgebung“ und „Umgebungsbeleuchtung“.
Ein Fuzzy Logic Controller ist hierbei für Kontrolle der Lautstärke der Betriebsgeräusche zuständig und nutzt als Kontexttypen Nutzeraktivität (Kontextatome: Bewegung, Laufen, Rennen) und Lautstärke der Umgebung (Kontextatome: leise, mäßig, laut). Die Zugehörigkeitsfunktionen der unscharfen Mengen sind dabei beispielhaft in Abbildung 5 dargestellt, wobei die Ausgabe die Lautstärke von Betriebsgeräuschen ist, z.B. Klingeltöne oder Mitteilungsalarme.

Zur manuellen Generierung der unscharfen Regelbasis - die später beschreibt wie der Controller Eingaben in Ausgaben umwandelt - werden Ein- und Ausgaben verwendet.
Ein weiterer Fuzzy Logic Controller ist für die Art der Darstellung von Informationen zuständig. Seine Eingaben sind dabei Nutzeraktivität und Umgebungsbeleuchtung (Kontextatome: dunkel, normal, hell). Die Ausgaben sind in diesem Fall Displaybeleuchtung und Textgröße, wobei letztere zusätzlich die Kompression der Informationen bedingt. Ein Beispiel für das Verhalten der Schriftgröße ist in Abbildung 6 zu sehen.

Hier ist zu erkennen, dass beispielsweise bei hoher Nutzeraktivität der Text groß ist, wohingegen er bei geringer Aktivität und heller Umgebungsbeleuchtung

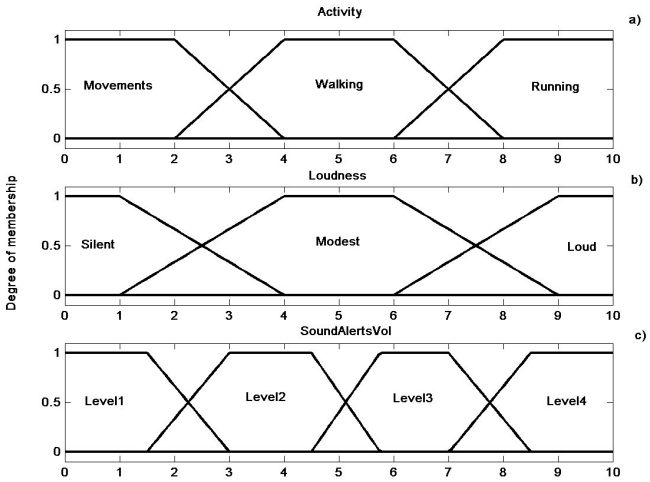


Abbildung 5. Zugehörigkeitsfunktionen zu Nutzeraktivität (a), Lautstärke der Umgebung (b) und Laustärke von Betriebsgeräuschen (c) [10]

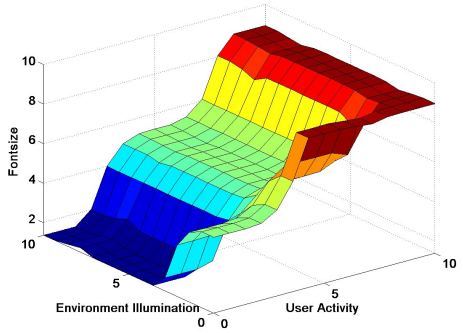


Abbildung 6. Verhalten der Schriftgröße [10]

klein dargestellt wird.
Die Bewertung des Systems wurde schließlich an einem realen Beispiel durchgeführt: Ein Nutzer fragt, während er sein Büro verlässt, den Busfahrplan ab und versucht einen Bus zu erreichen. Zuerst findet hierbei die Erstellung der Kontextatomvektoren statt, die dann als Eingaben für die Fuzzy Logic Controller dienen. Die Ausgaben beeinflussen dann das Verhalten des Geräts. Die im genannten Szenario gewonnen Erkenntnisse zeigten, dass es der Appilkation möglich war, sich mithilfe von Fuzzy Logic Controllern an den aktuellen Kontext anzupassen (z.B. Kontrollsignal für die Soundausgabe auf dem Minimum im Innern eines Gebäudes, Ansteigen, wenn der Nutzer beginnt zu laufen, etc.). Die Wahl der unscharfen Logik wird von den Autoren [10] damit begründet, dass im Gegensatz zu boolschen Regeln, welche Schwellwerte bilden, stetige Kontrollsignale ermöglicht werden. Zudem ist die Darstellung der Kontextinformationen in einem einheitlichen Format möglich. Darüber hinaus findet die vorgestellte Anwendung in der „realen Welt“ statt, womit unscharfe Kontexte und auch menschliche Betrachtungsweisen (laut, leise) einhergehen. Zudem wird hier beispielsweise bei der Repräsentation der Lautstärke davon Gebrauch gemacht, dass das Gesetz der ausgeschlossenen Mitte bei der unscharfen Logik nicht gilt.

3.2 Wahrscheinlichkeitstheorie

Anwendung 1: Probabilistischer Ansatz zur Aktivitätsüberwachung
West [11] stellt in seiner Arbeit einen Möglichkeit vor, wie normale Aktivitäten in einem „smart house“ repräsentiert werden können. Im Bereich der ubiquitären Informationssysteme ist dieser Ansatz beispielsweise hilfreich, um mithilfe von Sensoren und Ähnlichem älteren Menschen ein sicheres Wohnen zu ermöglichen. Das Hauptziel ist es, normale Aktivitäten zu erkennen und ein Maß einzuführen, das für solche unter einem Schwellwert bleibt, und diesen für abnormale Aktivitäten übersteigt. Da Abnormalität schwer zu modellieren ist - beispielsweise aufgrund des selteneren Auftretens - verfolgt West [11] den Ansatz, dass sich ein Gerät, sobald es eingeschaltet wird, in einem gefährlichen Zustand befindet. Dieser nimmt zu, je länger das Gerät unbeaufsichtigt ist. Zur Repräsentation dieser Zeit wird das Konzept der „anxiety“ eingeführt. Diese ist 0, wenn das Gerät eingeschaltet wird, und steigt, falls es nicht beaufsichtigt ist. Ein Gerät ist dabei dann als beaufsichtigt anzusehen, wenn entweder direkt mit diesem interagiert wird oder es aus der Nähe beobachtet wird. Diese beiden Fälle sollten dann dafür sorgen, dass die das Gerät betreffende „anxiety“ verringert wird und erst später wieder steigt. Eine weitere Möglichkeit zur Modellierung dieses Falls ist eine Funktion, ob entweder normal mit dem Gerät interagiert wird oder wie weit weg es ist (und somit die Distanz zu verwenden). Dies ist aufgrund der Tatsache sinnvoll, dass zum einen die Zeit, um ein Gerät zu erreichen, länger ist. Zum anderen kann bei entsprechendem Abstand nicht davon auszugegangen werden, dass dieses überwacht ist.
Als Beispiel für das bisher Genannte kann eine Küche dienen. Dabei sind die Verwendung des Herds und das Offenlassen des Kühlschranks potentiell gefährliche Situationen. Insofern sollte, wenn das jeweilige Ereignis eintritt, die „anxiety“

steigen, jedoch in Abhängigkeit davon, ob die Küche verlassen wird oder nicht. Beim Schließen bzw. Ausschalten wird die „anxiety“ zurückgesetzt. Weiterhin wäre es denkbar, einen Wert für die gesamte Küche zu berechnen, abhängig von den Werten aller gefährlichen Geräte. Bei mehreren unbeobachteten Geräten in gefährlichem Zustand muss dann eine Kombination berechnet werden.
Um schließlich festzustellen, was als normal und abnormal gelten soll, wird ein Lernansatz verwendet, da nur die jeweilige Person darin wirklich Einblick hat. Hierbei ist auch zu beachten, dass ein pessimistischer Ansatz gewählt wurde, dass also bei mehreren denkbaren Alternativen stets das schlechteste Ereignis angenommen wird.
Zur letztendlichen Definition der „anxiety“ werden die Geräte in zwei Klassen unterteilt: gefährliche und passive. Für Geräte gibt es nun einige statistische Repräsentationen:

- gefährliche Geräte
  - SID: Self Interaction Duration Model
    - $p_{SID}^{d_i}(t)$: Wahrscheinlichkeitsdichte der Zeitintervalle zwischen Interaktionen mit Gerät d_i
    - $P_{SID}^{d_i}(t_0, t)$: Wahrscheinlichkeit, ob mit dem Gerät zwischen dem Zeitpunkt der letzten Interaktion t_0 und der aktuellen Zeit t hätte interagiert werden sollen
  - IE: Interaction Event Model
    - $P_{IE}^{d_i,d_j}$: Wahrscheinlichkeit, dass mit einem anderen Gerät d_j interagiert wird, während sich d_i in einem gefährlichen Zustand befindet
- passive Geräte
  - IID: Inter Interaction Duration Model
    - $p_{IID}^{d_i,d_j}(t)$: Wahrscheinlichkeitsdichte der Zeitintervalle zwischen Interaktionen mit dem passiven Gerät d_j und danach mit dem gefährlichen Gerät d_i, vorausgesetzt d_i ist in einem gefährlichen Zustand
    - $P_{IID}^{d_i,d_j}(t_0, t)$: Wahrscheinlichkeit, dass zum Zeitpunkt t mit d_i hätte interagiert werden sollen, vorausgesetzt, dass zum Zeitpunkt t_0 mit d_j interagiert wurde

West [11] verdeutlicht diese Modelle mit einem Beispiel, wie es in Abbildung 7 zu sehen ist.

- In *Spalte A* ist ein Zeitraum von 20 Minuten angegeben.
- *Spalte B* enthält $p_{SID}^{d_i}(t)$ für den Herd mit einem Mittelwert bei etwa fünf Minuten und Interaktionsintervallen zwischen einer und zwölf Minuten.
- *Spalte D* zeigt $P_{SID}^{d_i}(0, t)$, welche von 0 auf ein Maximum bei 12 Minuten wächst.
- In *Spalte E* ist $p_{IID}^{d_i,d_j}(t-4)$ für den Schrank d_j zu sehen, vorausgesetzt der Herd d_i befindet sich in einem gefährlichen Zustand. Vor der dritten Minute sind keine Werte angegeben, da in der vierten Minute mit d_j interagiert wurde, nachdem d_i zuletzt beobachtet wurde. Weiterhin zeigt die Verteilung, dass die mittlere Zeit, mit dem Herd zu interagieren nachdem der Schrank geöffnet wurde, bei etwa sechs Minuten liegt.

	Stove			Cupboard					Overall
Time mins	PDF cts	CDF cts	CDF norm	PDF cts	CDF cts	CDF norm	1-G	1-G*0.7	D×I
A	B	C	D	E	F	G	H	I	J
0	0	0	0						0
1	5	5	0.030						0.030
2	10	15	0.092						0.09
3	15	30	0.18						0.18
4	20	50	0.30	0	0	0	1	0.3	0.30
5	25	75	0.46	0	0	0	1	0.3	0.46
6	24	99	0.61	0	0	0	1	0.3	0.61
7	20	119	0.73	2	2	0.05	0.94	0.34	0.25
8	15	134	0.82	5	7	0.20	0.79	0.44	0.36
9	12	146	0.90	10	17	0.5	0.5	0.65	0.58
10	8	154	0.95	10	27	0.79	0.20	0.85	0.81
11	5	159	0.98	5	32	0.94	0.05	0.95	0.94
12	3	162	1	2	34	1	0	1	1
13	0	162	1	0	34	1	0	1	1
14	0	162	1	0	34	1	0	1	1
15	0	162	1	0	34	1	0	1	1
16	0	162	1	0	34	1	0	1	1
17	0	162	1	0	34	1	0	1	1
18	0	162	1	0	34	1	0	1	1
19	0	162	1	0	34	1	0	1	1
20	0	162	1	0	34	1	0	1	1

Abbildung 7. einige beispielhafte Wahrscheinlichkeiten [11]

- *Spalte G* veranschaulicht $P_{IID}^{d_i,d_j}(4,t)$, welche von 0 bis zu einem Maximum bei 12 Minuten wächst.
- $P_{IE}^{d_i,d_j}$ ist auf 0.7 festgelegt und bezeichnet die Wahrscheinlichkeit mit dem Schrank zu interagieren, während sich der Herd in einem gefährlichen Zustand befindet.

Weiterhin soll noch realisiert werden, dass die „anxiety“ des Herds abnimmt, wenn mit dem Schrank interagiert wurde und im Folgenden wieder steigt. Diese Reduzierung soll jedoch nicht beibehalten werden, wenn nicht mit dem Herd interagiert wurde. Die Wahrscheinlichkeit des Herds wird also folgendermaßen angepasst:

$$S^{d_i,d_j}(t) = 1.0 - (P_{IE}^{d_i,d_j}(1.0 - P_{IID}^{d_i,d_j}(t)))$$

Dies führt zu:

$$P_{overall}^{d_i}(t) = P_{SID}^{d_i}(t - t_0) \times S^{d_i,d_j}(t - t_{e_j})$$

Hier wird nur die letzte Interaktion mit dem Gerät betrachtet, da nur diese relevant ist. Zudem soll sich die „anxiety“ für das gefährliche Gerät nicht weiter verringern, wenn mehrfach mit einem Gerät interagiert wurde.

Ein Beispiel ist in Abbildung 8 gezeigt: Die gesamte „anxiety“ wächst, bis mit dem Schrank interagiert wird. Daraufhin fällt diese und beginnt später wieder zu steigen. Der Wert 1.0 deutet schließlich an, dass keine Interaktion mit dem Herd

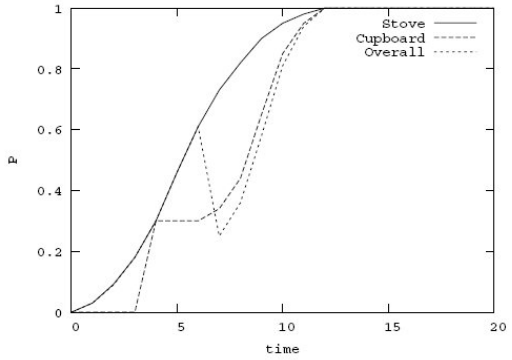


Abbildung 8. Graph der „anxiety“ des Herds (Spalte D), des Schrankes (Spalte I) und Gesamt-“anxiety“ (Spalte J) [11]

stattgefunden hat. Ein Vergleich der beiden Kurven (Herd- und Gesamt-„anxiety“) zeigt, dass die Interaktion mit dem Schrank die Zeit für einen bestimmten „anxiety“-Wert erhöht hat. Beispielsweise könnte 0.8 ein Wert sein, bei dem ein Alarm ausgegeben wird.
Nun sollte noch eine „anxiety“ für das gesamte Haus berechnet werden, vorausgesetzt jedes gefährliche Gerät verfügt über einen solchen Wert. Hierbei sollte nicht das Produkt verwendet werden, da dieser Wert geringer ist als das Maximum, z.B. sollte ein Wert von 0.5 bei zwei Geräten einen höheren Wert für die „anxiety“ des gesamten Hauses z.B. 0.75 als Resultat haben.
Bei einem Experiment wurde das System mit einer Reihe von möglichen Sequenzen des Frühstückzubereitens trainiert und schließlich wurden einige normale und abnormale Aktivitäten durchgeführt. Dabei zeigte sich, dass der genannte Ansatz für eindeutig definierte Szenarien in der Küche funktionierte.
In dieser Anwendung wurde die Wahrscheinlichkeitstheorie verwendet, um die Möglichkeit des Lernens vom Benutzer zu ermöglichen. Zudem wird eine feste Klasseneinteilung der Geräte vorgenommen und auch ein Schwellwert verwendet, ab dem eine gefährliche Situation eintritt. Überdies wird stets das „worst case“-Szenario angenommen, wenn zwischen zwei Alternativen nicht unterschieden werden kann. Dennoch handelt es sich eher um eine Anwendung aus der „realen Welt“ und auch der Begriff der „anxiety“ erinnert eher an ein menschliches Konzept. Dennoch wird gezeigt, dass auch in diesem Fall die Wahrscheinlichkeitstheorie anwendbar ist und passende Ergebnisse liefert.

Anwendung 2: Probabilistischer Lokationsdienst für Umgebungen mit drahtlosen Netzwerken

Castro [12] zeigt einen Ansatz, wie ein drahtloses Netzwerk genutzt werden kann, um mithilfe von Bayesnetzen eine Lokation durchzuführen. Diese bieten

die Möglichkeit des Lernens von Orten in einem Gebäude an und erlauben Flexibiltät aufgrund ihres modularen Aufbaus.
Zum Zweck der Lokation wurde die Applikation „Nibble“ entwickelt. Diese beruht auf einem „fusion service“- bestehend aus zwei Komponenten - um die Ortsbestimmung aufgrund von Signalstärkemessungen verschiedener Access Points durchzuführen. Die erste Komponente vereinigt und interpretiert Sensorinformationen, die zweite minimiert die Kosten des Datensammelns. Die Eingaben des „fusion service“ sind hierbei (probabilistische) Sensordaten, Ausgabe ist eine Wahrscheinlichkeitsverteilung einer Zufallsvariablen, die Kontextdaten beschreibt. Die Eigenschaften der Signale drahtloser Netzwerke (Rauschen, Unterbrechnungen) machen dieses Vorgehen notwendig.

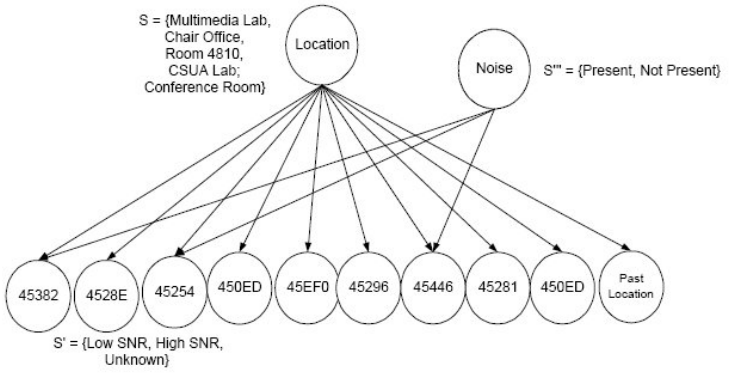


Abbildung 9. Beispielhaftes Bayesnetz zur Unterstützung der Generierung von Lokationsinformationen [12]

Im beschriebenen Ansatz werden Bayesnetze zur Interpretation von Sensorsignalen genutzt. Ein Beispiel ist in Abbildung 9 dargestellt. Es handelt sich um einen Baum mit gerichteten Kanten vom Wurzel- zu Endknoten, wobei Kanten eine Abhängigkeitsbeziehung andeuten. Der Wurzelknoten ist eine „Anfragevariable“ und repräsentiert $p(L)$, die a-priori Verteilung einer Menge von Orten $L = \{v_1, ..., v_i\}$. Die Endknoten beschreiben beobachtbare Variablen, welche $p(E|L)$ darstellen, wobei es sich um die bedingte Wahrscheinlichkeit handelt Werte $E = \{e_1, ..., e_n\}$ von einem Sensor zu erhalten, vorausgesetzt man befindet sich am Ort $v \in L$. Auf dieser Basis kann mithilfe der gemessenen Sensorwerte $R = \{e_1, ..., e_m\}$ (für Sensoren 1 bis m) $p(L|R)$ bestimmt werden, die a-posteriori Wahrscheinlichkeitsverteilung über L. Diese beschreibt die Wahrscheinlichkeit, dass man sich an einem beliebigen Ort v_i befindet, vorausgesetzt man erhält R. Anfragevariable kann jedoch nicht nur die Wurzel sein, sondern jeder beliebige Knoten. Als Beispiel könnte man nach $p(s_1 = e_1|L = v_3, s_2 = e_3)$ fragen, der

Wahrscheinlichkeit, dass Sensor s_1 den Wert e_1 liest, vorausgesetzt man befindet sich am Ort v_3 und Sensor s_2 liest e_3.
Zum Training des Bayesnetzes müssen die a-priori Verteilung $p(L)$ und die Verteilung $p(E|L)$ für jeden Access Point vorhanden sein. Dazu wurden in einem Experiment über einige Tage Signal-Rausch-Abstände (eingeteilt anhand fester Werte von kein=[0] bis hoch=(30,70)) an 14 verschiedenen Orten aufgezeichnet. $p(L)$ ist zu Beginn eine Gleichverteilung, jedoch sind für unterschiedliche Personen einige Orte wahrscheinlicher als andere. Diese persönlichen Eigenschaften können leicht durch Anpassen von $p(L)$ integriert werden.
Nibble, das als eine Komponente in einer größeren ortsbasierenden Anwendung geplant ist, wurde schließlich in zwei realen Szenarien getestet. Eines davon war ein Forschungslabor mit etwa 40 Büro- und einigen Konferenzräumen. Zur Darstellung der Lokationsdaten wurde eine Visualisierung verwendet, wobei die Wahrscheinlichkeiten in fünf Intervalle quantisiert wurden. Ein Nutzer wurde ausgewählt, welcher eine Reihe von Räumen benutzte. Dabei war es immer möglich drei Konferenzräume zu unterscheiden, da diese nicht direkt aneinander angrenzten. Bei direkt nebeneinander liegenden Büroräumen traten teilweise Fehler auf, was sich bei einem Büro Abstand auflöste.
Ein Vorteil der genannten Repräsentation ist die Möglichkeit, Nibble mit anderen Lokationssystemen mit kompatiblen probabilistischen Formulierungen zu verbinden. Eine mögliche Anwendung wäre, einigen Nutzern nur recht ungenaue Lokationsinformationen anzubieten z.B. auf Gebäudeebene, während andere beispielsweise eine Genauigkeit bis auf Raumebene erhalten könnten.
Eine zukünftige Möglichkeit, die Ergebnisse von Nibble zu verfeinern, sehen die Autoren [12] in der Verwendung eines Sensorboards, welches zusätzliche Daten wie Ausrichtung, Temperatur, ... messen kann. Sollten zwei Orte aufgrund der Daten des drahtlosen Netzwerks nicht eindeutig unterschieden werden können, wäre es möglich, dass beispielsweise die stets unterschiedliche Ausrichtung des Laptops eines Nutzers in beiden Räumen aufgrund deren Beschaffenheit Aufschluss geben könnte.
Die Nutzung der Wahrscheinlichkeitstheorie bietet in diesem Fall die Möglichkeit Bayesnetze als Mittel des Lernens einzusetzen. Insofern lässt sich auch deren Flexibilität ausnutzen und die Erweiterung um zusätzliche Dienste und andere Fähigkeiten (Ausnutzung und Hinzufügen weiterer Informationen) wird ermöglicht.

3.3 Hybrid

Auch Jihyung [13] beschreibt in seiner Arbeit eine Möglichkeit, Kontextinformationen für ubiquitäre Systeme zu nutzen. Allerdings wird hier ein Ansatz verfolgt, der sowohl unscharfe Logik nutzt als auch Wahrscheinlichkeitstheorie (in Form von Bayesnetzen).
Im hybriden Algorithmus wird ein Bayesnetz beispielsweise für das Lernen und Schließen von probabilistischen Beziehungen verwendet. Das hier verwendete Netz besteht aus fünf Knoten:

- Typ: beschreibt das Nutzerprofil

- Interesse: zeigt wie interessant die Arbeit ist
- mentaler Zustand: bewertet Nutzervorzüge bezüglich spezifischer Objekte
- Zeit: stellt dar, wie lange ein Nutzer vor einer Arbeit stand
- Präsentation: enthält spezifische anzeigbare Informationen

Die Tatsache jedoch, dass etwas für einen Nutzer interessant oder uninteressant sein könnte, bringt unscharfe Ereignisse ins System. Sowohl die Unschärfe der subjektiven Entscheidung als auch die Klassifikation machen einen unscharfen Ansatz nötig. Beispielsweise werden für den „Typ“-Knoten des Bayesnetzes unscharfe IF-THEN-Regeln verwendet und auch eine Klasseneinteilung geschieht hier mithilfe von Zugehörigkeitsfunktionen.
Der Algorithmus besteht letztendlich aus drei Stufen:

1. Quantifizierung der Kontextinformation für jeden Knoten im Bayesnetz mithilfe von unscharfer Logik
2. Berechnen der Knoten „mentaler Zustand“ und „Präsentation“
3. Entfernen des unscharfen Charakters des „Präsentation“-Knotens zur Darstellung der Informationen auf dem Bildschirm

Einen Überblick über das Aussehen des Algorithmus bietet Abbildung 10. Hierbei werden unscharfe Zugehörigkeitsfunktionen verwendet, um einen stochastischen Wert z.B. für den „Typ“-Knoten zu erhalten. Insofern werden von der unscharfen Logik die Eingaben für das Bayesnetz erstellt und nach der Bearbeitung diesem erneut mit dem Ziel der Informationsdarstellung übergeben.

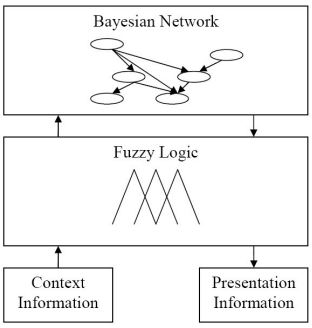


Abbildung 10. Verwendung von unscharfer Logik und Bayesnetzen im hybriden Algorithmus [13]

4 Fazit

In der vorliegenden Ausarbeitung wurde ein kurzer Überblick über unscharfe Logik präsentiert und diese anhand einiger Beispiele im Bereich der ubiquitären

Informationssysteme dargestellt. Weiterhin wurden auch Beispiele für die Nutzung von Wahrscheinlichkeitstheorie auf dem genannten Gebiet beschrieben. Es zeigte sich, dass weder unscharfe Logik noch Wahrscheinlichkeitstheorie überflüssig werden, wie Zadeh bestätigt [2]. Auch Almond [7] ist der Ansicht, dass die Theorien unterschiedliche Lösungen für verschiedene Klassen von Problemen ermöglichen, wobei er anmerkt, dass Statistiker lernen müssten, dass Ungenauigkeit nicht notwendigerweise schlecht ist. Darüber hinaus ist noch festzustellen, dass beide Theorien gemeinsam eingesetzt und somit ihre jeweiligen Vorteile vereinigt werden können, was sowohl Kandel [8] als auch Zadeh [2] feststellen. Klir [1] sieht als Hauptvorteile die Fähigkeit der unscharfen Mengentheorie zwischen mathematischen Modellen und der physikalischen Realität zu vermitteln und weiterhin Aussagen in natürlicher Sprache zu beschreiben. Dennoch gibt es weiterhin auch einige Skeptiker, die zwar gemäß Laviolette [6] einräumen, dass die unscharfe Mengentheorie nicht nutzlos ist, jedoch die Ansicht vertreten, dass keine Beispiele gefunden werden können, wo sie alleine nützlich wäre. Die Gegner der neuen Theorie sehen ihren Erfolg zum Teil im Versäumnis den möglichen Anwendern aufzuzeigen, wie deren Probleme mit Methoden der Wahrscheinlichkeitstheorie zu lösen sind. Dies zeigt, dass die Diskussion um die unscharfe Mengentheorie weiterhin nicht beendet ist, wenngleich sie bereits in einer Vielzahl von Anwendungen eingesetzt wurde und auch einige Befürworter hat.

Literatur

1. Klir, G. J.: „Fuzzy Logic - Unearthing its meaning and significance“, Potentials, IEEE, vol. 14, no. 4, pp. 10-15, Oct/Nov 1995
2. Zadeh, L. A.: „Discussion: Probability Theory and Fuzzy Logic Are Complementary Rather Than Competitive“, Technometrics, Vol. 37, No. 3 (Aug. 1995), pp. 271-276
3. Hanebeck, Uwe D.; Rößler, Patrick: „Skriptum zur Vorlesung Unscharfe Mengen“, Lehrstuhl für Intelligente Sensor Aktor Systeme, Institut für Technische Informatik, Universität Karlsruhe (TH), SS2005
4. Yen, J.: „Fuzzy Logic-A Modern Perspective“, IEEE Transactions on Knowledge and Data Engineering 11, 1 (Jan. 1999), 153-165
5. Bezdek, J.: „Fuzziness vs. Probability-Again!?“ IEEE Transactions on Fuzzy Systems, vol. 2, no. 1, pp. 1-3, February 1994
6. Laviolette, Michael; Seaman, Jr. , John W.; Barrett, J. Douglas; Woodall, William H.: „A Probabilistic and Statistical View of Fuzzy Methods“, Technometrics, Vol. 37, No. 3. (Aug., 1995), pp. 249-261
7. Almond, Russell G.: „Discussion: Fuzzy Logic: Better Science? Or better Engineering?“, Technometrics, Vol. 37, No. 3. (Aug., 1995), pp. 267-270
8. Kandel, Abraham; Martins, Alejandro; Pacheco, Roberto: „Discussion: On the Very Real Distinction between Fuzzy and Statistical Methods“, Technometrics, Vol. 37, No. 3. (Aug., 1995), pp. 276-281
9. Guarino, Donatella; Saffiotti, Alessandro: „Monitoring the State of a Ubiquitous Robotic System: A Fuzzy Logic Approach“, Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International , vol., no., pp.1-6, 23-26 July 2007
10. Mäntyjärvi, Jani; Seppänen, Tapio: „Adapting Applications in Mobile Terminals Using Fuzzy Context Information“ in „Mobile Human-Computer Interaction : 4th International Symposium, Mobile HCI 2002, Pisa, Italy, September 18-20, 2002. Proceedings“, vol. 2411/2002, pp. 383-404, 2002
11. West, G.; Greenhill, S.; Venkatesh, S., „A probabilistic approach to the anxious home for activity monitoring“, Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International, vol.1, no., pp. 335-340 Vol. 2, 26-28 July 2005
12. Castro, P., Chiu, P., Kremenek, T., and Muntz, R. R.: „A Probabilistic Room Location Service for Wireless Networked Environments“ in „Proceedings of the 3rd international Conference on Ubiquitous Computing“ (Atlanta, Georgia, USA, September 30 - October 02, 2001). G. D. Abowd, B. Brumitt, and S. A. Shafer, Eds. Lecture Notes In Computer Science, vol. 2201. Springer-Verlag, London, 18-34, 2001
13. Jihyung Park; Seungsoo Lee; Kiwon Yeom; Sungju Kim; Seokho Lee, „A context-aware system in ubiquitous environment: a research guide assistant“, Cybernetics and Intelligent Systems, 2004 IEEE Conference on, vol.2, no., pp.1213-1218, 2004

Die Grenzen Modellgetriebener Softwareentwicklung im Ubiquitous Computing

Youssef Ait Laydi

Email: s_aitlay@ira.uka.de
Betreuer: Till Riedel, TecO/Universität Karlsruhe

Einleitung

Anfang der 90er Jahre des vergangenen Jahrhunderts beschäftigte sich Marc Weiser, Wissenschaftler am XEROX Palo Alto Research Center (PARC), mit der Frage, wie die Computer für das 21. Jahrhundert aussehen könnten. Er prägte den Begriff des „Ubiquitous Computing“. Rechner werden immer unsichtbar und aufdringlich in Alltagsleben integriert sein. Sie werden dem Menschen ungeahnte neue Dienste zur Verfügung stellen, indem sie Information aus der Umgebung sammeln und diese automatisch zur Unterstützung des Menschen verarbeiten.

Mit neuen Arten von Geräten, wie Sensoren und Wearables, und neuen Formen der Information wird zwangsläufig auch die Menge der Daten rapide ansteigen. Die effiziente Verarbeitung der Daten durch ein zentralisiertes System wird kaum mehr möglich.

Modellgetriebene Entwicklungsansätze betonen die zentrale Rolle formaler Modelle und Modelloperationen im Lebenzyklus eines Softwaresystems und werden bereits in verschiedenen Anwendungsbereichen eingesetzt. Gegenstand dieses Beitrags ist die Fragestellung, inwiefern sich Konzepte modellgetriebener Entwicklungsansätze auf dem Bereich Ubiquitous Computing übertragen lassen.

Die Arbeit gliedert sich in drei Bereiche, der erste Teil gibt eine Einführung in Ubiquitous Computing, hier werden die wichtigsten Merkmale und Eigenschaften vorgestellt, außerdem wird hier drei Prototypen von Ubiquitous Computing betrachtet. Der zweite Teil erklärt die modellgetriebene Ansätze im allgemeinen, es wird in dieser Arbeit auf dem MDA- und MDSD-Ansätze fokussiert. Der dritte Teil stellt die modellgetriebene Ansätze in Ubiquitären Systemen vor, Hier wird eine Beispielanwendung dargestellt. In diesem Beispielanwendung werden die Arbeiten und Ergebnisse aus dem EMODE-Projekt vorgestellt, welches sich die Verbesserung der Effizienz der Entwicklung multimodaler, kontextsensitiver Anwendungen zum Ziel gesetzt hat, da die Kontext-Sensitivität eine der wichtigsten Anforderungen von Ubiquitous Computing ist. Dabei nutzt EMODE modellbasierte Entwicklung; wobei insbesondere die Integration der verschiedenen Entwicklungsschritte und eine durchgehende Werkzeugunterstützung betont werden.

1 Ubiquitous Computing

Dieses Kapitel beschreibt die Vision der ubiquitären Systeme und zählt Eigenschaften und Merkmale auf, die sich während der Forschung im Ubiquitous Computing herauskristallisiert haben. Es werden auch Schwierigkeiten und Herausforderungen bei der Umsetzung von Ubiquitous Computing dargestellt.

1.1 Einführung

Der Begriff des Ubiquitous Computing (Ubicomp) beschreibt eine Vision von zukünftigen Formen der Computernutzung, in der ein einzelner Nutzer auf natürliche Weise mit einer Vielzahl von Informationsgeräten interagiert, die unmerklich in seiner Umgebung eingebettet sind.

Die Vision des „Ubiquitous Computing“ (UC) beschrieb Marc Weiser 1991 wie folgt [Weis91]: „Die tiefgreifendsten Technologien sind die, die verschwinden. Sie verbinden sich mit den Strukturen des täglichen Lebens, bis sie von ihnen nicht mehr zu unterscheiden sind.“.

Anstelle von Desktop-Computern, deren Benutzung unsere volle Aufmerksamkeit fordert, treten nahezu unsichtbare Informationsgeräte, die uns in unserem Alltagsleben unterstützen. Ubiquitäre Informationsgeräte passen sich so an die Bedürfnisse der Nutzer an und nicht umgekehrt. [Kun05]

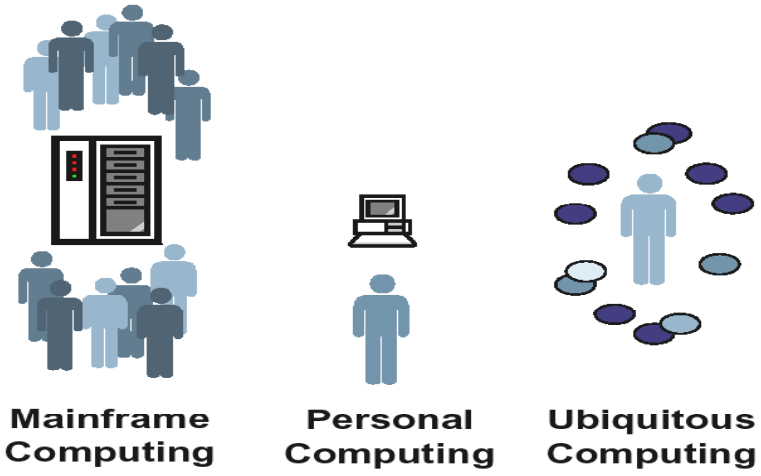


Fig. 1. Die drei Generationen der Computernutzung nach Weiser

Die erste Generation des *Mainframe Computing* ist gekennzeichnet durch eine hohe Anzahl von Benutzern pro Rechner. Computer sind in dieser Generation seltene Ressourcen, die meist von fachkundigem Personal oder Experten gemeinsam genutzt werden. Die Benutzung erfolgt dabei auf explizite Weise. Anfang der 1980er Jahre setzt sich das *Personal Computing* als zweite Generation durch, in der jeder Anwender einen persönlichen Computer benutzt. 1984 übersteigt die Anzahl der PC-

Benutzer erstmals die der Benutzer, die sich Computer teilen. Die Gruppe der Anwender ist nicht mehr auf Experten beschränkt, sondern umfasst auch unerfahrene Benutzer und die Benutzung ist direkt. In der neuen Generation des *Ubiquitous Computing* ist jede Person im Alltag von einer Vielzahl unterschiedlicher Rechner umgeben und die Benutzung erfolgt auf implizite Weise. Dies bedeutet dass der Nutzer hauptsächlich seine Aufgaben erledigt und sich nicht bewusst ist, einen Computer zu verwenden.

Das Hauptziel von Ubiquitous computing ist die unsichtbare Einbettung von Geräten in die reale, physikalische Altagswelt. [Kun05]

1.2 Zugrunde liegende Technologien

Als Mark Weiser 1991 die Idee des Ubiquitous Computing erstmals formulierte, griff er zur Realisierung seiner Vision auf viele damals verfügbare Technologien zurück: Eingebettete Systeme bildeten die Grundlage seiner Tab Computer, drahtlose Kommunikation ermöglichte deren Vernetzung. Aus dem Bereich der Verteilten Systeme kommen Methoden zum Zusammenwirken verschiedener Informationssysteme zur Erfüllung einer Gesamtfunktion. Auch heute greift man bei der Beschäftigung mit Ubiquitous Computing auf die aktuellen technologischen Fortschritte in diesen Bereichen zurück und nutzt diese zum Entwurf von unbewusst nutzbaren Informationsanwendungen. Die treibenden Technologien sind dabei im Wesentlichen dieselben geblieben:

•Eingebettete Systeme ermöglichen die Integration von Informationstechnik in Objekte der realen Welt

• Middleware-Technologien vereinfachen die Nutzung verteilter Systeme

• Die große Entwicklung des World Wide Web (WWW) zum globalen Informationsnetz. Dadurch wurde dem inzwischen fast allgegenwärtig verfügbaren Zugriff auf beliebige Informationen eine wichtige Infrastruktur für ubiquitäre Informationssysteme geschaffen. [Mpr06]

• Die große Verbreitung der Mobilfunk-Kommunikation. Innerhalb weniger Jahre wandelten sich Mobiltelefone vom teuren Kommunikationsmedium für wenige, meist professionelle Nutzer zum fast für jeden verfügbaren Alltagsgegenstand. Mobiltelefone haben heute meist mehr Fähigkeiten als die von Weiser skizzierten Pad-Computer, und ihre Nutzung ist für die meisten Menschen heute selbstverständlich geworden.

1.3 Anforderungen von Ubiquitous Computing

Physikalische Einbettung

Die Einbettung strebt danach die Informationssysteme für den Benutzer unsichtbar zu machen, das ist ein erster Schritt auf dem Weg zu den unbewussten nutzbaren Anwendungen und findet sich bereits deutlicher in dem Begriff, disappearing

Computer. Wenn das Gerät selbst mobil ist, dann hat es in der Regel einige Einschränkungen in Bezug auf die Physikalische Sicht. Diese physikalischen Einschränkungen begrenzen die Ressourcen, wie z.B. Batterie-Kapazität, Bildschirmgröße, Bandbreite, und so weiter [Gur02]. Einen Besonderer Fall der Einbettung ist die Wearable Computing, dies bedeutet die Integration von Informationstechnischen Funktionen in Kleidung ein.

Interoperabilität

Die Interoperabilität repräsentiert einen sehr wichtigen Aspekt in Ubiquitous Computing. Die wichtigste Frage lautet: Wie stellt man die flexible Interaktion von Geräten, Programme und Diensten im Ubiquitous Computing sicher? Diese Vielzahl an Komponenten und ihre (teilweise) Mobilität verdeutlichen die Notwendigkeit einer möglichst hohen Interoperabilität. [Kun05]

Unter Interoperabilität ist die „Fähigkeit unabhängiger, heterogener Systeme“ zu verstehen, „möglichst nahtlos zusammen zu arbeiten, um Informationen auf effiziente und verwertbare Art und Weise auszutauschen bzw. dem Benutzer zur Verfügung zu stellen, ohne dass dazu gesonderte Absprachen Zwischen den Systemen notwendig sind.“[Wik05]

Im Ubicomp kommt der Aspekt der spontanen Interoperabilität in wechselnden Umgebungen hinzu. Eine spontane Interaktion liegt vor, wenn eine Komponente mit anderen Komponenten zusammenwirkt, deren Identität und deren Funktion sich über die Zeit ändern können. Bedingt durch den mobilen Charakter von Ubicomp-Systemen, gehört die Änderung der Umgebung und der Einsatzbedingungen zu deren normalem Betrieb. Ad-hoc-Vernetzung ist eine sehr wichtige Anforderung in der Interoperabilität, Ad-hoc-Netzwerke bilden also die Grundlage einer spontanen Interaktion, welche eine Kommunikation zwischen zwei Geräten ohne vorherige Kenntnis voneinander ermöglichen. Trotzdem müssen sich die vernetzten Geräte auch auf einer semantischen Ebene verständigen können. [GW01]

Bei der Betrachtung von Vernetzung und spontaner Interaktion dürfen natürlich Sicherheit und Datenschutz nicht vernachlässigt werden. Interoperabilität kann positive und negative Auswirkungen haben, so dass ein Abwägen zwischen Funktionalität und Missbrauchsrisiken erforderlich ist.

Interaktion mit der physikalischen Welt

Eine wichtige Forderung an ubiquitäre Systeme ist die nach Interaktion mit der realen Welt. In klassischen Systemen ist der Benutzer die einzige Schnittstelle zur realen Welt. Er gibt Informationen über eine Benutzerschnittstelle ein und nimmt verarbeitete Informationen entgegen. Ubiquitäre Systeme interagieren zusätzlich mit der physikalischen Welt, indem sie Informationen aus ihrer Umgebung über Sensoren aufnehmen und unter Umständen auch auf ihre Umgebung z.B. über Steuergrößen einwirken.

Die Interaktion zwischen Informationsanwendungen und realer Welt ermöglichen neuartige Formen der Benutzerinteraktion. Informationssysteme werden in die Lage versetzt, sich der aktuellen Situation ihrer Umgebung anzupassen.

Die Integration mit der physikalischen Welt über Sensoren ist auch eine Voraussetzung für eine weitere Eigenschaft ubiquitärer Systeme, die Kontext-Sensitivität.

Kontext-Sensitivität

Um die Kontext-Sensitivität zu verstehen, muss man zuerst den Begriff Kontext definieren. Kontext ist jede Information, die verwendet werden kann, um die Situation eines Wesens zu charakterisieren. Ein Wesen ist eine Person, ein Ort, oder ein Objekt, die relevant zur Interaktion zwischen einem Benutzer und einer Anwendung berücksichtigt werden, einschließlich den Benutzer und die Anwendungen selbst. Unter den Begriff von Kontext-Sensitivität versteht man die Anpassung der Umgebung an die Bedürfnisse der Benutzer. Dies geschieht, wenn die Sensoren die Informationen aus der Umgebung erfassen, in diesem Zusammenhang mit Sensorinformationen steht der Kontext. Ein Kontext beschreibt in ubiquitären Systemen den Zustand, der durch die Auswertung von Informationen Über die Umgebung eines Benutzers oder Systems entsteht [Dey01]. Meistens benutzt man den Begriff von Kontext um die Physikalische Umgebung zu beschreiben (physical context) [Lan99], Andere Ansätze betrachten zusätzlich einen situationsbezogenen Kontext (situation context), welches die Situation beschreibt, in der sich der Benutzer oder das System gerade befinden. [Aeh02]

In Abbildung 2 wird ein Anwendungsmodell mit Kontextbezug vorgestellt. Hierbei wird Kontext angesehen als eine Anwendung und was den Benutzer umgibt. Dies gilt in der Informationswelt, als auch in der physikalischen Welt.

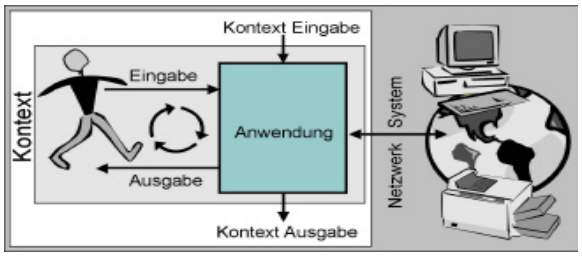


Fig. 2. Anwendungsmodell für interaktive Anwendungen mit Kontextbezug

Sehr einfache Formen der Kontext-Nutzung lassen sich in GUI beobachten, z.B. Kontextmenüs. In Anwendungen, die ihren Kontext kennen, ermöglichen implizite Interaktion. Unter impliziter Interaktion werden Handlungen des Benutzers verstanden, die nicht primär zur Interaktion mit dem Computer durchgeführt werden, aber vom Computer als Eingaben interpretiert werden. Ein einfaches Beispiel ist ein Benutzer, der durch die Tür in seine Wohnung geht. Dies ist eine Handlung, die er nicht durchführt, um mit einem Computer zu interagieren, die aber sehr wohl von einem Computer als Eingabe interpretiert werden kann. Diese Diskussion und der damit verbundene Paradigmenwechsel von expliziter zu impliziter Interaktion wurden in [Sch00] vertieft.

Aufgabenorientierte Geräte
Mit Aufgabenorientierte Geräte sind gemeint, dass alle verschiedenen Aufgaben auf verschiedene Geräte aufteilen lassen. Dieses Konzept erschwert uns allerdings die Verknüpfung verschiedener Tätigkeiten.

Eine sehr ähnliche Entwicklung kann heute bei den sich noch entwickelnden Smartphones beobachtet werden. Sie vereinen die Funktionen eines Mobiltelefons, eines Organizers und oft auch einer Digitalkamera und einer mobilen Spielekonsole, die Akzeptanz solcher Multifunktionsgeräte muss sich allerdings erst noch unter Beweis stellen. Dem versucht das Ubiquitous Computing eine Vielzahl aufgabenspezifischem, vernetzt Informationsgeräte entgegenzusetzen. Solche Aufgaben-orientierte Geräte werden nach Norman als *information appliance* bezeichnet [Nor96].
Durch die Aufteilung der verschiedenen Aufgaben auf verschiedene Geräte wird allerdings auch eine Verknüpfung verschiedener Tätigkeiten erschwert. Diese Dezentralisierung kann allerdings durch eine Vernetzung der Geräte und gegenseitige Dienstnutzung kompensiert werden. So kann eine vernetzte Digitalkamera in Verbindung mit einem Access-Point ebenso zum Verschicken eines Bildes per Email verwendet werden wie ein Mobiltelefon mit integrierter Kamera.

2 Die Modellgetriebener Ansätze

MDSD steht für Model-Driven Software Development und ist Vorgehensmodell zur Software-Entwicklung, bei dem zentral Modelle (und weniger zentral Code) zur Beschreibung von Software benutzt werden. MDA ist eine spezielle, von der OMG standardisierte Ausprägung des MDSD. Sie unterscheidet sich von MDA in dem Plattformbegriff. Letztendlich betrachtet die MDSD viel stärker den Entwicklungsprozess. [SV05]

Bei MDSD werden Modelle als abstrakt und formal zugleich angesehen. Sie sind abstrakt, da implementierungsspezifische Details weggelassen werden. Die Modelle zeigen ausschließlich Eigenschaften, die das Verständnis für die Problemstellung fördern. Formal sind diese Modelle, da sie sich an einer Domain Specific Language (DSL) orientieren. Bei MDSD stehen die benutzten Modelle immer im Kontext eines abgegrenzten Problemraums, der Domäne. Die Konzepte dieser Domäne werden mit einer für sie spezifischen Sprache beschrieben, der DSL. Eine DSL kann zum Beispiel durch ein UML-Profil abgebildet werden.

Eine Domänenspezifische Sprache oder Domain Specific Language kurz DSL benötigt neben einem Metamodell noch eine Definition der (dynamischen) Semantik. Mit anderen Worten der Modellierer muss die Konstrukte der Sprache verstehen. Dazu gibt es zwei Möglichkeiten: entweder ist die Bedeutung eines Konstrukts intuitiv verständliche oder es existiert eine natürliche sprachliche Beschreibung. [Ben06]

In diesem Kapitel werden die Grundlagen von MDA und MDSD vorgestellt, dabei werden die Ziele, die verschiedene Konzepte und basistechnlogien von beiden Ansätzen dargestellt.

2.1 Modell-Driven Architektur (MDA)

Die Kern Idee der MDA besteht darin, dass die Spezifische Softwarekomponenten unabhängig von ihrer technischen Umsetzung beschrieben werden sollten, dazu unterscheidet man zwei verschiedene Modelle, Plattform-unabhängigen Modelle sowie Plattform-spezifischen Modelle. Dabei sichert die MDA durch eine klare Trennung von Abstraktionsschichten bei der Modellierung von Systemen die Wiederverwendbarkeit und Langlebigkeit der Modelle. Durch die automatische Generierung von Quellcode aus den Modellen wird zudem der Automatisierungsgrad der Entwicklung erhöht und somit Fehlerquellen minimiert.

2.2 Ziele der MDA

Im Folgenden sind die verschiedene Ziele der MDA dargestellt:[Gpr06]

Effiziente Softwareentwicklung: Durch den Ansatz von MDA werden die Software-Prozess schneller, und die Qualität der Software durch den Einsatz von Entwurf muster und Codegeneratoren im Entwicklungsprozess gestiegen. Durch den Einsatz von wieder verwendbaren Template und Architekturmuster reduziert der Einsatz der MDA die Fehleranfälligkeit.

Portierbarkeit: Im Kontext der MDA ermöglicht die Definition eines plattformunabhängigen Modell die Implementierung auf unterschiedlichen Zielplattformen und Zieltechnologien, wodurch eine höhere Portabilität der Entwicklung realisiert werden kann. Die Langlebigkeit wird eminent erhöht.

Integration und Interoperabilität : Die MDA als Rahmenstandard der OMG vereint eine Reihe relevanter Standards wie UML und XMI und erleichtert durch die Nutzung von Protokollen zur verteilten Kommunikation die Systemintegration, d.h. die plattformübergreifende Zusammenarbeit von Softwaresystemen. Dadurch wird die Zusammenarbeit unterschiedlicher Systeme ermöglicht. Zusätzlich wird die Wiederverwendbarkeit bereits bestehender Komponenten erleichtert und dadurch die Produktivität durch Vermeidung unnötiger Redundanzen in der Softwareentwicklung gesteigert.

Zentrale Fehlerbehebung: Fehler, die im schematischen Codeanteil auftreten, können an einer Stelle - den Transformationsvorschriften – behoben werden. Danach sind sie in allen generierten Codeteilen beseitigt und treten auch in zukünftigen Anwendungen nicht mehr auf. Darüber hinaus bieten die Transformationsvorschriften Gelegenheit querschnittlich verteilte Implementierungsaspekte gebündelt umzusetzen.

Bessere Handhabbarkeit aufgrund höherer Abstraktion: Mit den Modellierungssprachen soll der Programmierer auf ein abstrakteres Niveau arbeiten. Die Unabhängigkeit zwischen die Spezifikation und die technische Umsetzung zielt auf eine bessere Wartbarkeit durch Trennung der Verantwortlichkeiten ab.

2.3 Konzepte von MDA

Die wesentlichen Konzepte (Modelle) für die MDA sind im Folgenden erläutert:[Gpr06] [BB02]

- **Computation Independent Model (CIM):** Mit Hilfe des Computation Independent Models (CIM) wird die Geschäfts-oder Domänensicht des zu entwickelnden Softwaresystems modelliert. Das CIM ist im Rahmen der MDA der Modelltyp mit der höchsten Abstraktionsebene. Das CIM ist unabhängig von technischen Aspekten der Implementierung,
- **Platform Independent Model (PIM):** Das PIM der MDA bezieht sich auf die Unabhängigkeit von technologiespezifischen (J2EE, CORBA) sowie Hersteller-gebundenen (Microsoft, WebSphere, and WebLogic) Plattformen. Dabei ist es wichtig nur die rein fachlichen Aspekte zu betrachten und zu modellieren, so dass das entwickelte Modell immer gültig ist, obwohl überhaupt keine Software entwickelt wird. Dadurch besteht die Möglichkeit, je nach Anforderungen, das PIM (automatisch durch entsprechende Übersetzer) in ein PSM zu transformieren.

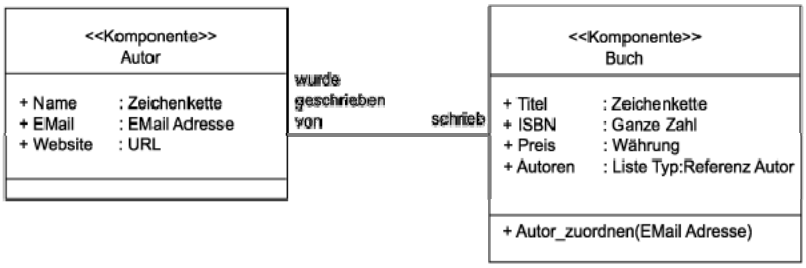


Fig. 3: Beispiel PIM

Exemplarisch könnte man in diesem Kontext z.B. einen Sortieralgorithmus, der mittels eines PIM beschrieben ist, an eine konkrete Ausführungsplattform z.B. Java binden. Dadurch, dass diese Bindung größtenteils automatisch durchgeführt werden kann, entstehen geringere Kosten, wenn die Plattformbindung anschließend aufgehoben werden muss, z.B. zugunsten einer performanteren C++ Plattform.

- **Platform Description Model (PDM):** PDMs sind Metamodelle, die die Zielplattform des Systems beschreiben, sie helfen dabei die Transformation von PIM zu PSM zu erleichtern. Damit dieser Vorgang in möglichst großen Teilen automatisch geschehen kann wurde im Rahmen der MOF für die Plattformprofile (auf der Meta-ebene) ein Meta-Modell entwickelt, welches eine konzeptionelle Basis für das Mapping liefert.
- **Platform Specific Model (PSM):** Das Ergebnis der Modelltransformation wird in der Terminologie der MDA als PSM (Platform Specifc Model) bezeichnet. Als Beispiel, ist die Umsetzung eines Online-Shops. Das System benötigt zur Speicherung von Informationen hinsichtlich Benutzer, Artikel, Kreditkarten usw. Man kann für diesen Zweck auf die Benutzung von Oracle

Datenbank entscheiden. Um diese Arbeit zu realisieren, man benötigt ein Oracle SQL Dialekt um diese Konzepte in einem relationalen Modell zu formulieren. (z.B. das Konzept eines Benutzers). Diese Oracle spezifischen relationalen Modell ist ein Beispiel für eine Plattform Spezifisch Modell.

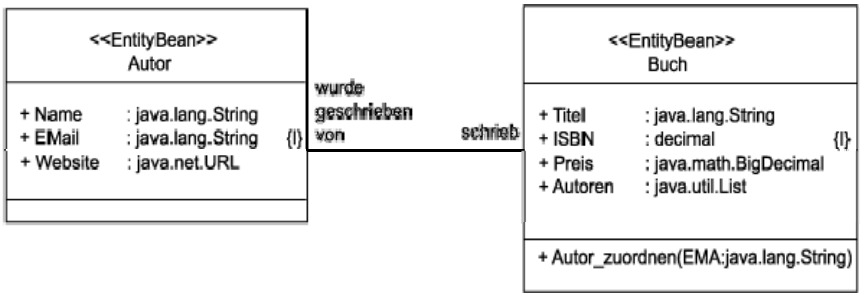


Fig. 4: Beispiel PSM

2.4 Basistechnologien der MDA

Die MDA vereint eine Vielzahl von standardisierten Technologien der OMG. Nur mit Hilfe dieser Technologien lässt sich der prinzipielle Ansatz der MDA umsetzen. Die wichtigsten Basistechnologien in Rahmen der MDA sind UML und MOF. UML bietet eine Vielzahl Eigenschaften die ideal zur Erfüllung der an eine gute Modellierungssprache gestellten Anforderungen passen:[BB02]

- UML: Die Unified Modeling Language (UML), die von der Object Management Group(OMG) – einem Industriekonsortium, in dem fast alle Softwarefirmen vertreten sind – standardisiert wird, ist deshalb ein wesentlicher Fortschritt. Die UML basierte zunächst auf einer Reihe von objektorientierten Modellierungsansätzen für die Entwicklung von Softwaresystemen mit komplexen Datenstrukturen. Die Standardisierung und kontinuierliche Weiterentwicklung durch die OMG sichert der UML eine breite Akzeptanz, insbesondere auch durch Werkzeughersteller, welche die für eine derartige Sprache notwendigen Editoren, Animations-, Analyse-, Transformations- und Generierungswerkzeuge entwickeln.
- MOF: Die Meta Object Facility (MOF) ist ein Standard der OMG zur Definition von Metamodelle-Sprachen. MOF lassen sich in zwei große Teile gliedern: (a) Die Definition und Wartung von Metadaten basierend auf einer vier Metadaten Ebene und (b) Spezifikationen der Schnittstellen um innerhalb einer verteilten Umgebung auf Metadaten zuzugreifen.
- UML Profile: UML-Profile sind der Standardmechanismus zur Erweiterung des Sprachumfangs der UML, um sie an spezifische Einsatzbedingungen anzupassen.

Sie bilden im Rahmen der MDA die Grundlage für die automatische Modelltransformation. Mit deren Hilfe und die entsprechenden Transformationsregeln wird die Abbildung eines MDA Modell auf eine gegebene Plattform eindeutig definiert, d.h. es kann z.B. Ein PIM in ein für eine definierte Zielplattform entsprechendes PSM umgewandelt werden.

3 Die Modellgetriebener Ansätze in Ubiquitous Computing

Ubiquitous Computing strebt nach der Unterstützung der Nutzer in seinem täglichen Aufgaben, indem sie ihn relevante und implizite Dienste zur Verfügung stellt. Solche Nutzerunterstützung führt dazu die Einsatz von Anwendungen in den verschiedensten Orten, Einstellungen und Bedingungen.

3.1 MDA Ansätze in Ubiquitous Computing

Um die Wiederverwendbarkeit der Software zu erhöhen und die Anpassung der Applikationen zu einer neuen Entwicklungsumgebung zu erleichtern, soll man MDA zu einem Ubicomp Umgebung anpassen. MDA entkoppelt die Applikationen von ihrem Ausführungsplattform via Modell Abstraktion und Modell Transformation. Ein Applikation wird zunächst durch eine Platform Independent Model (PIM) beschrieben, enthält aber keine Informationen über die endgültige Ausführungsplattform und Middleware. Dieses Modell ist mehrmals durch aufeinanderfolgende Transformationen verfeinert, jede Transformation integriert neue Einschränkungen wie Persistenz und Redundanz. Sobald die PIM genug detailliert ist, wird eine Ausführungsplattform gewählt dann ist die PIM in Plattform-specific Model PSM transformiert. Dies PSM wird dann wiederum transformiert um einen Code zu generieren und die Applikation zu kompilieren.

3.2 Anpassung der MDA zu Ubiquitous Computing

Um dies Konzept zu erklären, betrachten wir eine Applikation die einen älteren erkrankenden(z.B. Demenz) Mensch bei seinem Alltäglichen Leben hilft. Diese Anwendung kann die Bewegungen und alle Aktivitäten dieser Person überwachen um Änderungen der Gewohnheiten zu detektieren. Beispielsweise betrachten wir eine Applikation die überprüft wie die Person den Tisch zum Abendessen stellt, wenn diese Person ein wesentliches Element (z.B. Messer oder Gabel)vergisst, löst ein visueller oder akustischer Alarm an.

Das ursprüngliche PIM sollte das wichtigste Applikationslogik und geometrische Logik enthalten, in unserem Beispiel definiert man die Applikationslogik als eine Überwachungskamera die regelmäßig die Konfiguration auf dem Tisch überprüft und informiert der Nutzer wenn es eine falsche Aufstellung des Tischs entdeckt. Beim geometrisches Logik geht um die Positionen des einzelnen Objektes, der Tisch ist ein

Bereich, in dem mehrere Objekte gleichzeitig drauf stehen müssen, wie gabel und Messer

Das PIM wird mehrmals transformiert, jede Transformation modifiziert das PIM um Einschränkungen zu integrieren, hier definiert man zwei arten von Einschränkungen, Benutzer und geometrische Einschränkungen. (Sie sind technologieunabhangig) Benutzereinschränkungen definieren die Behinderungen der Nutzer wie Blindheit und Taubheit. Geometrische Einschränkungen beschreiben die geometrische Konfiguration des Ortes in dem die Applikation angewendet werden soll. Sobald Nutzer und geometrische Einschränkungen integriert sind, die PIM wird in ein PSM transformiert um die Umgebungs- und Technologiebedingungen zu integrieren. Umgebungsbedingungen beschreiben die Umgebung, unter denen die Anwendung verwendet werden soll: Home / Fabrik, Innen- und Außenbereich, laut, dunkel. Technologiebedingungen beschreiben die Grenzen der Technik, wie Bildschirmgröße für einen Benutzer oder die Genauigkeit für einen Lokationssensor. [PVBB]

3.3 Softwareanforderungen der Ubiquitous Computing

Benutzer wollen normalerweise mit Computer interagieren, im Fall der Ubiquitous Computing ist anders, Ubicomp ist in der physikalischen Umgebung für der Benutzer eingebettet und nahtlos in der täglichen Aufgaben integriert. Diese Vorstellung führt uns dazu die wichtigsten Merkmale und Herausforderungen der Ubiquitous Computing zu definieren. [Mpr06]

Anhand des realen Szenarios in [Gur02] kann man verschiedene Eigenschaften des Ubicomp darstellen, drei wesentliche Merkmale werden hier hervorgehoben:

- **Task Dynamism:** Ubiquitous Computing sind überall und zu jeder Zeit erhältlich, deswegen muss man sich dem Dynamismus der Benutzersumgebung und darauf resultierender Unsicherheit anpassen, sodass Benutzer ihre ziele oder Aktionen glücklicherweise ändern können. z.B. Eine plötzliche Änderung des Flugzieles
- **Geräteheterogenität und Ressourceneinschränkungen:** Wenn wir unseren heutigen realen Welt betrachten dann können wir feststellen dass das Konzept von Allgegenwärtigkeit der Computer Technologie erreichbar ist, dies kann durch die Mobilität und die Miniaturisierung der Geräte erzielt werden. Wenn das Gerät mobile ist, dann hat es üblicherweise Einschränkungen, wie Batteriestandzeit, Bandbreite, Bildschirmgröße usw. Der zweite Aspekt ist die Hardware Fähigkeiten der Geräte und die zur Verfügung gestellte Softwaredienste[GrDA]
- **Computing in sozialen Umwelt:** Die Einführung eines Ubiquitous Computing Umgebung hat einen enormen Einfluss auf die sozialen Umwelt, z.B. Das Einsetzen von Sensoren überall in unserer Umgebung hat unwiderruflich einen Einfluss auf die Umweltstruktur, egal ob sie unauffällig sind oder nicht. Stellen Sie sich vor dass Ihre Gebäude mit allen Arten von Sensoren, die Information zu einem Ubiquitous Computing System

bereitstellen, ausgestattet. Wollen Sie also dass die Nachbarschaftspolizeireviere das Raum, das Sie z.Z. besetzen, überwachen?

3.4 Softwareherausforderungen der Ubiquitous Computing

Mit Hilfe der Charakteristiken in vorherigen Unterkapitel können wir etliche Herausforderungen für die Ubiquitous Computing hinsichtlich Softwareentwicklung definieren:[Gur02]

- **Semantische Modellierung:** Viele kontextbezogene Anwendungen werden auf komplexen räumlichen und hochdynamischen Modellen der Realwelt beruhen, deren konsistente und effiziente Verwaltung eine große Herausforderung darstellt. Mittels High-Level semantic Model kann man die Präferenzen des Benutzers beschreiben, Ontologie[Gru93] kann verwendet werden, um Aufgabe der Benutzersumgebung zu beschreiben, sowie ihre Ziele, Außerdem ermöglicht uns gute Beschreibungen der Geräte und ihre entsprechende Fähigkeiten und Anwendungsfeld wie man den Benutzer so gut wie möglich in irgendwelchen Kontext unterstützen kann.
- **Aufbau der Software-Infrastruktur:** Die erste Herausforderung ist für das System festzustellen, welche Benutzeraufgaben zu einem Benutzer in einem bestimmten Kontext am relevantesten sind. Anwendungen müssen angemessene Funktionalität anbieten, selbst wenn Netzverbindung zeitweilig oder nicht erreichbar ist, und müssen die Ausfällen entdecken. Schließlich muss die Software-Infrastruktur in der Lage sein , verschiedener Komponenten zu Gesamtsystemen zu integrieren, mit heute verfügbaren Komponenten wie Kamera-basierten Erkennungssystemen, Satellitennavigation oder Smartphones ließe sich theoretisch eine große Anzahl von ubiquitären Anwendungen realisieren. Die Komponenten-Integration in wechselnden, unvorhergesehen Konfigurationen mit wechselnden Nutzern bleibt allerdings ein schwierige Aufgabe.
- **Selbstorganisation:** Durch die hohe Komplexität solcher Systeme müssen sie weitgehend selbstorganisierend sein, d.h. die Konfiguration, Fehlerbehebung und Optimierung müssen ohne Zutun des Benutzers erfolgen. Außerdem müssen sie in der Lage sein, sich automatisch an sich ändernde System- und Realweltumgebungen anzupassen.

3.5 Fallstudie: Modellgetriebene Entwicklung multimodaler, kontextsensitiver Anwendungen, EMODE

Die Entwicklung multimodaler, kontextsensitiver Anwendungen gewinnt zunehmend an Interesse. Jedoch stellen diese höheren Anforderungen an den Softwareentwicklungsprozess. In diesem Kapitel werden die Arbeiten und Ergebnisse aus dem EMODE-Projekt vorgestellt, welches sich die Verbesserung der Effizienz der Entwicklung multimodaler, kontextsensitiver Anwendungen zum Ziel gesetzt hat.

Dabei nutzt EMODE modellbasierte Entwicklung; wobei insbesondere die Integration der verschiedenen Entwicklungsschritte und eine durchgehende Werkzeugunterstützung betont werden.

EMODE baut auf einen modellgetriebenen Softwareentwicklungsansatz. EMODE setzt sich gezielt die Verbesserung der Effizienz der Entwicklung multimodaler, kontextsensitiver Anwendungen zum Ziel. Dabei grenzt sich EMODE insbesondere von anderen Ansätzen durch den Anspruch auf umfangreichere Unterstützung und Integration der einzelnen Entwicklungsschritte ab.

Dafür wurde eine Werkzeugkette entwickelt, welche die Entwicklung von Benutzerschnittstellen mit Hilfe der aus der Literatur bekannten Modelle unterstützt. Doch während in anderen Ansätzen die Entwicklung des konkreten Benutzerschnittstelle (CUI-Modells) das Ziel ist, verwendet EMODE alle Modelle, inbegriffen Kontext, zur Code-Generierung. Der generierte Code schließlich wird in der EMODE Laufzeit ausgeführt. Transformationensbeschreibungen zur Unterstützung des Entwicklers sind in anderen Ansätzen oft in Werkzeugen kodiert (Teresa, usiXML) oder in nicht standardisierten Sprachen (usiXML) ausgedrückt. Dem wirkt EMODE entgegen, indem alle Transformationen in QVT, einem Standard der OMG, beschrieben sind. Nachfolgend wird der Ansatz von EMODE anhand der entwickelten Methodik vorgestellt. Des Weiteren wird eine Werkzeugkette beschrieben, die den EMODE-Ansatz implementiert.

EMODE-Methodik

Die EMODE Methodik beschreibt einen Ansatz für die Entwicklung multimodaler, kontextsensitiver Anwendungen. Sie definiert eine Reihe von Entwicklungsphasen, verschiedene Modelle, die während dieser Phasen entwickelt werden, und Transformationen, die den Entwicklungsprozess unterstützen (siehe Fig.5). Eine bemerkbare Idee liegt hier besonders auf der Einbeziehung von Anwendern, UI Designern und Anwendungsentwicklern, deren Interessen und Fähigkeiten während des Entwicklungsprozesses berücksichtigt und integriert werden müssen.

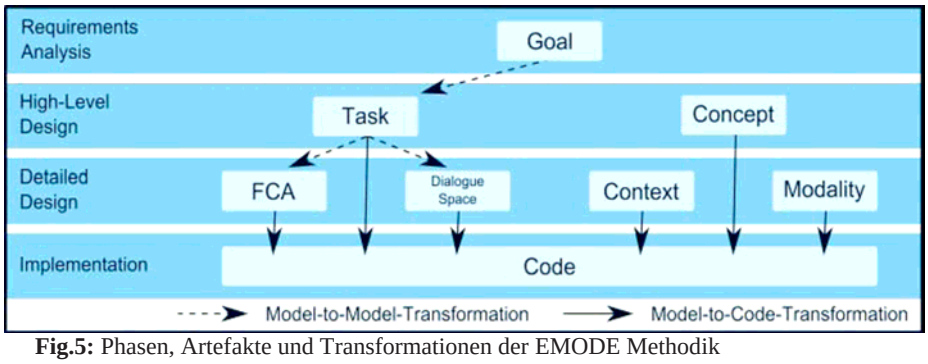


Fig.5: Phasen, Artefakte und Transformationen der EMODE Methodik

Es werden vier verschiedene Phasen Während des Entwicklungsprozesses durchlaufen, in denen unterschiedliche Modelle erstellt oder durch eine Transformation gewonnen und verfeinert werden. In der ersten Phase („Requirements Analysis“) wird das Goal-Modell erstellt, welches die funktionalen Anforderungen

der Anwendung beschreibt. Die funktionalen Anforderungen und deren Relationen untereinander bilden die Grundlage für das Task-Modell. In der zweiten Phase „High-Level Design“ wird das Task-Modell bearbeitet. Darin wird der Ablauf der zu modellierenden Anwendung in Form von System-Tasks und Interaction-Tasks erfasst, sowie deren Abhängigkeiten zueinander durch Kontrollflüsse abgebildet. Außerdem werden in dieser Phase die benötigten Anwendungskonzepte im Concept-Modell modelliert und in das Task-Modell integriert. Das fertige Task-Modell stellt den gesamten Ablauf der Anwendung dar. Aus dem Task-Modell werden in der dritten Phase„Detailed Design“ das DialogueSpace-Modell und das FCA-Modell erstellt. Das UI wird von einem UI-Experten modelliert, indem im DialogueSpace-Modell eine Initiale, durch Transformation generierte UI Beschreibung schrittweise für verschiedene Modalitäten verfeinert wird.

Ein Anwendungsentwickler kann über das FCA-Modell Anwendungslogik anbinden. Besonders wichtig ist hier die Beziehung der Modelle Task, DialogueSpace und FCA. Veränderungen (z. B. das Entfernen eines Tasks) werden an die jeweils anderen Modelle propagiert um für Konsistenz zu sorgen. Weiter werden in dieser Phase Kontextprovider (Dienste, die der Anwendung Kontextinformation verfügbar machen) modelliert, und festgelegt, welche Modalitäten die Anwendung unterstützt. In der Implementation Phase wird der Quellcode der Anwendung erstellt. Dies geschieht durch eine Modell-zu-Code Transformation bei der fast alle Modelle als Informationsquelle dienen (siehe Fig.5). Die Implementierung der Anwendungslogik in die vorgenerierten Klassen muss von Hand geschehen. Für den Modellierungsprozess sind vor allem die Modelle Task, DialogueSpace, FCA und Concept wichtig. Um mehr Funktionalität zu erreichen (z. B. Kontextsensitivität) sind allerdings weitere Modelle (z. B. Context-Modell) notwendig. Auf Basis der EMODE Methodik wurde eine Werkzeugkette entwickelt, die nun kurz beschrieben wird.

Bereitgestellte Werkzeuge

Die EMODE-Entwicklungsumgebung stellt eine Reihe von Werkzeugen bereit, mit deren Hilfe multimodale, adaptive Anwendungen auf Basis der EMODE-Methodik erstellt werden können. Das Editieren von Modellen in EMODE erfolgt durch grafische Editoren, in gewohnter Drag’n’Drop-Manier. Die Modelle werden in einem zentralen Modellrepository abgespeichert. Modell-zu-Modell und Modell-zu-Code Transformationen werden von integrierten Transformationsengines ausgeführt. Alle Editoren, das Modellrepository und Transformationen sind in einer Umgebung, Eclipse, integriert, welche auch die Bearbeitung des generierten Codes erlaubt. Somit kann der gesamte Entwicklungsprozess nahtlos durchgeführt werden.

Beispielanwendung: Szenariobeschreibung

Eine Aufgabe der Instandhaltung ist die schnellstmögliche Reparatur aufgetretener Störungen. Die genutzte Infrastruktur besteht aus einer Monitoring-Komponente, einem zentralen Server und einer Menge mobiler Endgeräte, welche die Instandhaltungsmitarbeiter bei ihrer Arbeit unterstützen und deren Kommunikation mit dem Server ermöglichen. Sobald die Monitoring-Komponente Störungen in der Produktionshalle feststellt, wird ein Reparaturauftrag an den Server übermittelt.

Infolgedessen werden alle Instandhaltungsmitarbeiter über die aufgetretene Störung informiert. Akzeptiert ein Arbeiter den Reparaturauftrag, ändert sich der Auftragsstatus und die Reparaturarbeiten werden begonnen. Während der Reparatur wird der Arbeiter durch die Instandhaltungsanwendung über potentielle Gefahrenquellen, benötigte Utensilien, etc. unterrichtet. Dabei wird die Ausgabemodalität des UIs in Abhängigkeit vom Umfeld (z. B. Lärmpegel, Helligkeit) bestimmt.

Modellierungsschritte

Den Ausgangspunkt der Modellierung bildet das Anlegen des Task-Modells, welches im Wesentlichen das Verhalten der Applikation abbildet. Für die Task-Modellierung in EMODE wurde neben Elementen aus UML-Aktivitätsdiagrammen (Kontrollflüsse, Objektflüsse, Startknoten, etc.) das Task-Konzept aus dem ConcurTask-Tree-Ansatz [Pat97] übernommen. Demzufolge beinhaltet das Task-Modell insbesondere System- und Interaction-Tasks. System-Tasks kapseln Aufgaben, die ausschließlich vom System erbracht werden (z. B. Datenbankabfragen). Im Gegensatz dazu verlangen Interaction-Tasks eine Mensch-Maschine-Kommunikation (z. B. Texteingaben). Unterschieden wird in EMODE zwischen Kontroll- und Datenabhängigkeiten („Flüssen“) zwischen Tasks. Mit Hilfe von Kontrollflüssen wird gesteuert, in welcher Reihenfolge Tasks aktiv sind. Kontrollflüsse können durch Entscheidungs- und Vereinigungsknoten gesteuert werden.

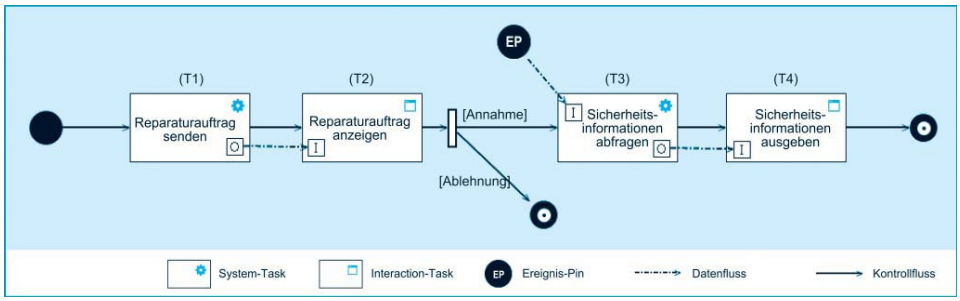


Fig.6: Task-Modell der Beispielanwendung

Wichtig für die späteren Schritte sind auch Datenflüsse, in Fig.6 gestrichelt dargestellt. Mit ihnen wird modelliert welche Daten zwischen welchen Tasks übergeben werden. So folgt im Szenario auf das Senden eines Reparaturauftrags des Systems (System-Task T1) die Anzeige des Auftrags beim Nutzer (Interaction-Task T2). Dabei wird zum Einen die Kontrolle von T1 an T2 weitergegeben, zum Anderen auch der Reparaturauftrag (modelliert im Concept-Modell). Datenübergabepunkte an Tasks (Pins) sind hierbei typisiert durch modellierte Elemente aus dem Concept-Modell. Die Integration von Kontextdaten ist an System-Task T3 in Fig.6 zu sehen. Hierbei wird der Kontextdienst nach den aktuellen Umgebungsbedingungen gefragt. Dazu wurde im Context Modell ein Provider modelliert, der die benötigte Information (z.B. Umgebungslautstärke) ableiten kann. Im Concept-Modell wurde vorher das Konzept der benötigten Information (also die Umgebungslautstärke) modelliert. Die

Abfrage an den Kontextdienst ist im Element EP (Ereignis Pin) gekapselt. Die Benennung kommt daher, dass der Kontextdienst Daten auch asynchron (d. h. nicht nur bei Task-Start) an einen Task übermitteln kann. Auch die asynchrone Datenübermittlung zwischen Tasks wird von EMODE unterstützt. Nicht zu sehen in Fig.5 ist die Möglichkeit Hierarchien zu erstellen – hierfür kann ein Task vom Typ „abstrakt“ genutzt und in einem weiteren Modell verfeinert werden. Mit Hilfe von Verzweigungsknoten kann der Kontrollfluss gespalten und mit Vereinigungsknoten wieder vereint werden. Im zweiten Entwicklungsschritt wird das FCA-Modell durch Modelltransformationen aus dem Task-Modell abgeleitet. Ein FCA-Call wird dabei als Realisierung eines System-Tasks (d. h. Aufruf von Anwendungslogik) verstanden. Die Anwendungslogik liegt in einer Methode (FCA-Methode), welche ggf. eine vom Call abweichende Signatur haben kann, z. B. bei Nutzung einer Methode durch mehrere verschiedene Calls. Das FCA Modell verknüpft somit System-Tasks mit Methoden der Anwendungslogik und beschreibt deren Übersetzung ineinander durch Zuordnung der Parameter. Das Resultat der Modell-zu-Modell- Transformation ist das generierte FCA Modell in Fig.7.

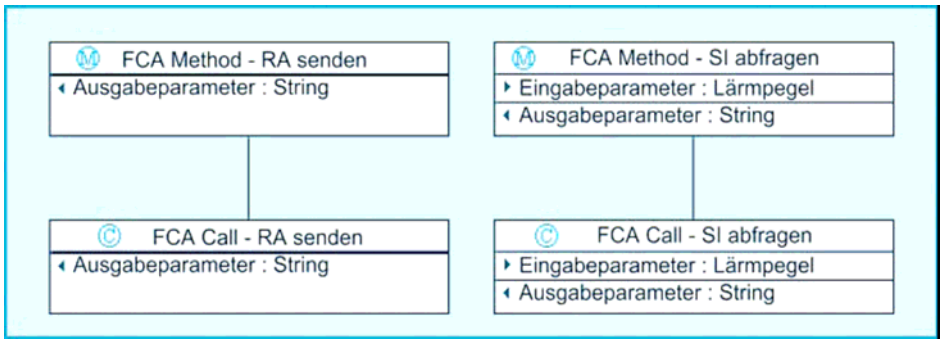


Fig.7: FCA-Modell der Beispielanwendung

Dabei werden System- Tasks in FCA-Calls überführt und Task mit Call verknüpft, sowie dazu passende FCA-Methods generiert und mit den Calls verknüpft. Typisierte Pins im Task-Modell dienen als Quelle für Parameter im FCA-Modell. Beispielsweise wird im FCA-Modell in Fig.7 für den System-Task T1 der FCA-Call „RA senden“ erzeugt und auf Grund des Ausgabe-Pins an T1 ein passender Parameter an den Call gehängt. Dazu wird eine entsprechende FCA-Methode generiert, welche die Grundlage für die Code-Generierung des Methoden Rumpfes bildet. Im gewählten Beispiel ist jedoch eine weitere Anpassung des FCA-Modells nicht notwendig.

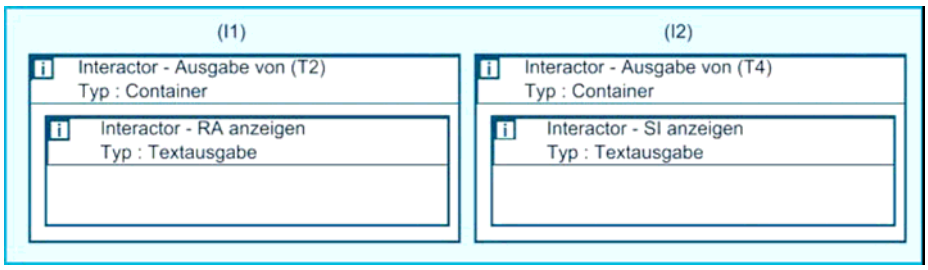


Fig.8: DialogueSpace-Modell der Beispielanwendung

Das ebenfalls aus der Modelltransformation hervorgegangene DialogueSpace-Modell in Fig.9 dient der Spezifikation multimodaler Benutzerschnittstellen. Dafür werden Interactor-Elemente (zu sehen in Bild 4) bereitgestellt, welche über ein Klasse-Instanz-Konzept einen Typ zugewiesen bekommen können (z. B. Container, Textausgabe, …). Bei der Transformation von Task nach DialogueSpace werden im Wesentlichen Interaction-Tasks auf AUI-Interactors abgebildet und für Pins an den Interaction-Tasks geschachtelte Interaktoren erzeugt. So wird z. B. für Task T2 in Fig.6 der Interactor „Ausgabe von (T2)“ erzeugt. Dieser wiederum enthält den Interactor „RA anzeigen“, welcher auf Grund des Eingabepins „I“ an T2 erzeugt wurde. Im Beispiel in Fig.9 wurden die Namen zur besseren Illustration geändert, sowie die Typen gesetzt.

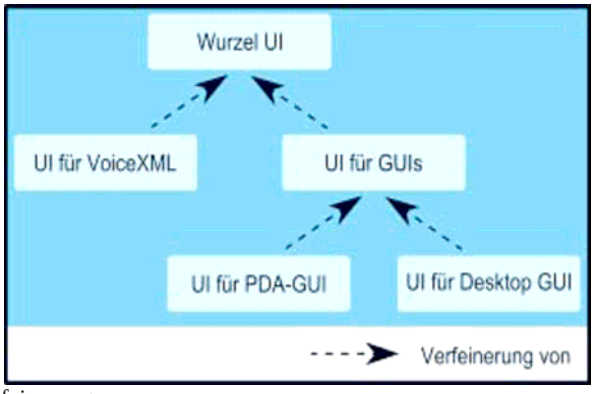


Fig.10: UI Verfeinerungsprozess

In Fig.10 ist die Verfeinerung für unterschiedliche Modalitäten illustriert. Dabei wird ein Wurzel UI (dies ist i.A. das initial generierte und mit dem Task-Modell verknüpfte) für Sprache und GUI verfeinert. Bei der Verfeinerung werden die Typen und Eigenschaften der UI Elemente verändert, z. B. wird ein abstraktes „Wähle 1-aus-N“ in „Combo-Box“ geändert. Der Übergang von „UI für GUIs“ zu „UI für PDA-GUI“ kann z. B. die Größenangaben der GUI Elemente verfeinern, welche im „UI für GUIs“ evtl. noch nicht gemacht wurden. Somit ist die schrittweise Anpassung an die gewünschten Modalitäten möglich. Information, welche für verschiedenen Zielmodelle gleich ist (in Fig.10 z. B. die Hintergrundfarbe für PDA und Desktop) kann auf passendem Abstraktionsniveau (im Beispiel „UI für GUIs“) gegeben werden und nicht separat für jedes Zielmodell. Nach Abschluss der Modellierungsphase, kann aus den Modellen ausführbarer Code erzeugt werden. Die im DialogueSpace- Modell gefassten Benutzerschnittstellen können in verschiedene Zielsprachen (aktuell D3ML [Göb06], Java AWT) transformiert werden. Daneben wird das Task-Modell in Java-Klassen überführt, welche durch die EMODE Laufzeitumgebung interpretiert werden. Für die im Context-Modell beschriebenen Kontextprovider zur Ableitung höherwertigen Kontexts werden Methodenrümpfe inkl. Anbindungslogik an den EMODE Kontextdienst erzeugt. Bevor die entworfene Beispielapplikation ausgeführt werden kann, ist ein letzter Schritt notwendig: die Implementierung der

Anwendungslogik. Zu diesem Zweck sind die Java-Klassen, welche aus dem FCA- und Context-Modell hervorgegangen sind, mit Platzhaltern für die Implementierung versehen, wie in Bild 6 gezeigt.

```
public class EmodeApplicationFca {

    public String getMaintenanceOrder() {

        String returnValue = "default";
        /* begin business logic */

        /* end business logic */
        return returnValue;
    }
}
```

Fig.11: Aus FCA-Methode generierte Klasse

Ist die Anwendungslogik und ggf. die Kontextprovider implementiert, erfolgt das Deployment der Anwendung. Das Deployment beschränkt sich dabei auf das Hinzufügen speziell entwickelter Laufzeitbibliotheken zu den kompilierten Java-Klassen.

4 Zusammenfassung

Als Mark Weiser seine Vision in seinem Artikel[8] beschrieben hat:

Ubiquitous computing is about interconnected hardware and software that are so ubiquitous that no one notices their presence. This will enable people to focus on their tasks and on interacting with other people.

Diese weitreichende Vision ist noch so weit von unserem Ziel. Aber Die dynamische Entwicklung in diesen Gebieten geht ungebremst weiter, die Auswirkungen ihrer technischen Errungenschaften betreffen daher immer größere Teilbereiche des täglichen Lebens. Damit wird auch deutlich, dass das 21. Jahrhundert wohl weniger, wie frühere populäre Zukunftsprognosen es nahe legten, durch Mondkolonien, Unterwasserstädte und Atomautos geprägt sein wird, die alle den Einsatz großer (und damit teurer und nur im Rahmen eines gesellschaftlichen Konsenses realisierbarer) Technikstrukturen voraussetzen, als vielmehr durch die Anwendung kleinster33 und damit quasi unsichtbarer, aber gerade dadurch leicht replizierbarer und verbreitbarer Technik.

Die kommenden Jahre werden also neben dem Nachweis der technischen Realisierbarkeit ubiquitärer Informationsanwendungen auch zeigen müssen, ob die sozialen, rechtlichen und wirtschaftlichen Herausforderungen gelöst werden können oder ob Ubiquitous Computing eine Vision ist, die bei einem Großteil der Benutzer auf Ablehnung stößt.

Die MDA ist ein noch relativ junger Standard. Es ist daher nicht verwunderlich, wenn es in der praktischen Anwendung noch viele Schwierigkeiten gibt. Zum einen ist die Umsetzung des Standards in entsprechenden Werkzeugen noch nicht

überzeugend und zum anderen gibt es noch eine Reihe konzeptioneller Aspekte rund um die MDA-basierte Softwareentwicklung, die noch nicht ausreichend geklärt erscheinen. Einige dieser Aspekte sind im folgendem aufgeführt und erläutert:

Stabilität des Datenmodels: Änderungen am Datenmodell können großen Aufwand nach sich ziehen, der bei der Änderung des Modells nicht sofort sichtbar ist.
Entfremdung der Entwicklers vom Code: Der generierte Code ist für den Entwickler nur noch schwer verständlich und damit schwer nachvollziehbar.
Unzulänglichkeiten in den Tools: Die Erwartungshaltungen der Entwickler sind derzeit nur schwer zu erfüllen.
In dem letzten Kapitel wurden Arbeiten aus dem EMODE-Projekt vorgestellt, das den Entwickler bei der Erstellung multimodaler, kontextsensitiver Anwendungen unterstützt. EMODE unterscheidet sich insbesondere durch den hohen Integrationsgrad der Entwicklungsumgebung und im Umfang der Unterstützung von anderen Ansätzen. Im Gegensatz zu anderen Ansätzen kann mit dieser integrierten Umgebung eine komplette, lauffähige Anwendung, inkl. Kontextanbindung und multimodalen UIs erstellt und ausgeführt werden. Dabei erlaubt EMODE nicht nur die Nutzung einfacher Kontextinformation, sondern auch die Ableitung höherwertigen Kontextwissens.

EMODE unterscheidet sich insbesondere durch den hohen Integrationsgrad der Entwicklungsumgebung und im Umfang der Unterstützung von anderen Ansätzen. Im Gegensatz zu anderen Ansätzen kann mit dieser integrierten Umgebung eine komplette, lauffähige Anwendung, inkl. Kontextanbindung und multimodalen UIs erstellt und ausgeführt werden. Dabei erlaubt EMODE nicht nur die Nutzung einfacher Kontextinformation, sondern auch die Ableitung höherwertigen Kontextwissens. Mit der Auflösung der strikten Trennung von AUI und CUI in das DialogueSpace-Modell, stellt EMODE eine Möglichkeit bereit, UIs auf beliebigem Abstraktionsniveau zu beschreiben, und diese schrittweise, über mehr als eine Stufe zu verfeinern. Damit wird auch die große Variabilität im Abstraktionslevel von möglichen Sprachen für Zielmodalitäten adressiert. Essentiell ist dabei der Verzicht auf UI Elementtypen im Metamodell. Stattdessen kann EMODE durch ein Klasse-Instanz Konzept mit weiteren UI Elementtypen ohne Metamodelländerung erweitert werden. Zurzeit wird auch an einer Ontologie-Unterstützung der UIVerfeinerung gearbeitet – sie soll UI Elementtypen bzw. deren Eigenschaften miteinander in Beziehung setzen und einen höheren Automatisierungsgrad für Transformationen ermöglichen. Verschiedene Test-Anwendungen wurden bereits mit EMODE erfolgreich gebaut. Erste Ergebnisse sind:

• EMODE bietet eine umfangreiche Unterstützung des Entwicklers - von der UI Entwicklung bis hin zur Integration der Anwendungslogik.

• Die Modell-zu-Modell-Transformationen eignen sich um Modellierungs- Aufgaben zu automatisieren. Zu beachten ist jedoch die Komplexität deklarativer Transformations-Beschreibung.

• Die Integration der verschiedenen Editoren wirkt sich positiv auf die Entwicklungsgeschwindigkeit aus. Insbesondere werden Änderungen an einem

Modell, welche ein anderes Modell beeinflussen, sofort sichtbar und ermöglichen somit kurze Iterationszyklen. Verschiedene Aspekte, wie z. B. Verteilung von UIs, Benutzer-Modelle, Gruppen-Modelle, werden in Zukunft stärker in der Softwareentwicklung berücksichtigt werden müssen und erfordern ein entsprechendes methodisches Vorgehen. Modell-getriebene Ansätze sind ein vielversprechender Weg um dieser zunehmenden Komplexität zu begegnen.

Literaturverzeichnis

[Wei91] Mark Weiser. The Computer for the 21st Century. In Scientific American, pages 94–104, September 1991.

[Nor96] Norman, A. D.: "The Invisible Computer", Cambridge, Massachusetts; MIT Press. 1998

[Bag05] Faruk Bagci. Reflektive mobile Agenten in ubiquitären Systemen Doktors der Naturwissenschaften. Dezember 2005

[Kun05] Christophe Kunz. Ubiquitous Healthcare: Anwendung ubiquitärer Informationstechnologien im Telemonitoring. DISSERTATION Oktober 2005

[Wei96] Mark Weiser, John Seely Brown: "The coming age of calm technology", Xerox PARC, Oktober 1996

[Wik05] Wikipedia, s.v.: http://de.wikipedia.org/wiki/Interoperabilität Juni 2005

[Göb06] Göbel, S.; Hartmann, F.; Kadner, K.; Pohl, C.: A Device-Independent Multimodal Mark-up Language. *GI Jahrestagung* **2** (2006) 170–177.

[Gur02] Guruduth Banavar and Abraham Bernstein. Software Infrastructur and Design Challenges for Ubiquitous Computing Applications. Dezember 2002/Vol. 45, No 12 Communication the ACM

[GW01] K. Green, J.C. Wilson: "Future power sources for mobil communications",Electronics and Communication Engineering Journal, Februar. 2001

[Lan99] lan Dix, Tom Rodden, Nigel Davies, Jonathan Trevor,Adrian Friday, and Kevin Palfreyman. Exploiting space and location as a design framework for interactive mobile systems. In ACM Transaction on Computer-Human Interaction (TOCHI), pages 285–321, 1999

[Aeh02] Gregory D. Abowd, Maria Ebling, Guerney Hund, Hui Lei, and Hans-Werner Gellersen. Context-Aware Computing. In IEEE Pervasive Computing, Juli 2002

[Tau06] TAUCIS. Bizer Johann, Dingel Kai, Fabian Benjamin, Günther Oliver, Hansen Markus ,Klafft Michael Möller Jan, Spiekermann Sarah Juli 2006

[Fär05] G. Färber. Manuskript zur Vorlesung Eingebettete Systeme. Sommersemester2005

[Gpr06] V.Gruhn, D.Peiper, C.Röttgers. MDA Effektives Software-Engineering mit UML 2 und Eclipse. Springer-Verlag 2006

[SV05] Thomas Stahl, Markus Völter. Modellgetriebene Software-Entwicklung dpunkt.verlag 2005

[BB02] Jean Bézivin & Xavier Blanc. MDA : VERS UN IMPORTANT CHANGEMENT DE PARADIGME EN GENIE LOGICIEL Juli 2002

[PVBB] Julien Pauty, Stefan Van Baelen, Yolande Berbers. Adapting Model-Driven Architecture to Ubiquitous Computing K.U. Leuven, Department of Computer Science

[Dey01] A. Dey: Understanding and using context. Personal and ubiquitous computing, volume , number 1, 2001

[Mpr06] Mario Prinz. Modellgetriebene Entwicklung ubiquitärer Web-Anwendungen. Diplomarbeit Wien, 24. April 2006.

[GrDA] Gregory D. Abowd Software Engineering Issues for Ubiquitous Computing College of Computing & GVU Center Georgia Institute of Technology

[Gru93] Gruber, T. R. A translation approach to portable ontologies. *Knowledge Acquisition* *5* 2 (1993), 199–220.

[Dgh02] Davies, N.; Gellersen, H.W.: „Beyond Prototypes:Challenges in Deploying Ubiquitous Systems“, IEEE Pervasive Computing, 2002-1 pp. 26-35. IEEE, 2002

[Sch00] *Schmidt, A.*, Implicit Human Computer Interaction Through Context. Personal Technologies Volume 4(2&3), June 2000. pp191-199.

[Ben06] Benedikt Weismann Architekturzentrierte Modellgetriebene Softwareentwicklung-Fallbeispiel und Evaluierung Magisterarbeit zur Erlangung des akademischen Grades eines Magister der Sozial- und Wirtschaftswissenschaften. Wien, 15. September 2006

[Wei93] Some Computer Science Issues in Ubiquitous Computing, Marc Weiser, 23. März 1993

[Pat97] Paternò, F.; Mancini, C.; Meniconi, S.: Concur-TaskTrees: A Diagrammatic Notation for Specifying Task Models. In: *INTERACT ‘97: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*. London: Chapman & Hall,

Modellgetriebene Entwicklung von Ubiquitären Informationsumgebungen

Daniel Wildschut, wildschu@teco.edu
Betreuer Patrik Spieß, SAP Research, CEC Karlsruhe, patrik.spiess@sap.com

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)

1 Einleitung

Ubiquitous Computing, auch bekannt unter den Namen Pervasive Computing, ambient intelligence oder everyware ist das Gebiet der allgegenwärtigen Computer, eingebettet in Alltagsgegenstände. Ubicomp Applikationen verbinden typischerweise viele verschiedene Geräte in einer großen Gesamtapplikation. Ein Teilbereich davon ist die Entwicklung für drahtlose Sensornetzwerke, welche herkömmlicherweise durch Bottom-Up Design erfolgt, wobei für jeden Sensorknoten und jedes eingebundene Gerät einzeln die Funktion spezifiziert wird. Für große Netze von eventuelle heterogenen Geräten ist dies jedoch kaum praktikabel. Ein anderer Ansatz versucht daher durch Analyse und Verteilung eines globalen Programms die Gesamtheit der Geräte auf einmal zu programmieren. Es soll untersucht werden, ob es hier Ansätze gibt, die eine solche Programmierung auf hohem Abstraktionsniveau erlauben. Dies wird typischerweise durch Modellierung des Gesamtsystems erreicht. Der erste Teil stellt eine Auswahl der bestehenden Modellierungsansätze im Ubicompbereich vor. Der zweite Teil zeigt dann anhand von Beispielen den aktuellen Zustand der Makroprogrammierung von Sensornetzwerken auf. Schlussendlich werden die Erkenntnisse zusammengefasst und untersucht ob die eingangs gestellte Anforderung befriedigt werden kann.

2 Modellgetriebene Entwicklungssysteme

2.1 Allgemein

Die Entwicklung komplexer Applikationen erfordert eigentlich immer eine aufwändigen Designprozess, der die verschiedenen Anforderungen mit einbezieht. Zur Unterstützung dieser Entwicklungsmethode gibt es die modellgetriebene Entwicklung, in der durch Erstellen und Transformieren von Modellen die Applikation schrittweise entworfen wird. So könnte im ersten Schritt eine Anforderungsanalyse bestimmen was für Funktionen das System beherrschen muss. Daraus wird dann im nächsten Schritt die Struktur der benötigten Komponenten gewonnen, für die dann in einem weiteren Schritt die Implementation bereitgestellt wird. Indem in jedem Schritt des Entwicklungsprozesses ein passendes Modell des gewünschten Systems verfeinert wird, wird ein durchgehend hoher Abstraktionsgrad erreicht, so dass spezifische Implementationsdetails erst am Ende wichtig werden.

2.2 Model driven architecture

Einer der wichtigsten modellgetriebenen Ansätze ist Model-driven architecture[1] (MDA), der von der Object Management Group 2001 publiziert wurde. In MDA

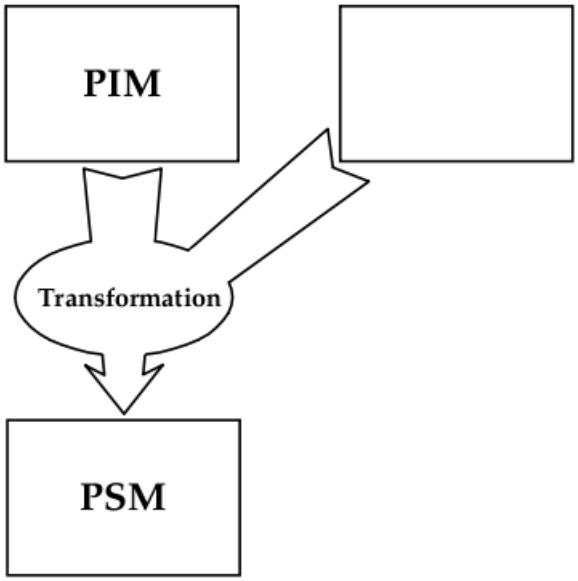


Abb. 1. MDA Modelltransformation

wird zuerst mithilfe einer Modellierungssprache wie UML oder einer domainspezifischen Sprache ein sogenanntes *platform independent model*(PIM) erstellt. Dies repräsentiert das zu entwerfende System unabhängig von der Plattform, für die es entwickelt wird. Daraus generiert man dann mithilfe von einer Modelltransformation(vgl. Abb. 1) ein sogenanntes *platform specific model*(PSM). Aus dem PSM wird schlussendlich die eigentliche Implementation generiert. Diese Unterteilung in verschiedene Schritte hat unter anderem den Vorteil, dass aus einem PIM verschiedene PSMs generiert werden können. Es ist daher in Theorie möglich eine Applikation ohne größeren Aufwand auf verschiedene Architekturen zu portieren.
Für die Entwicklung von ubiquitären Anwendungen ist die gegebene MDA nur bedingt geeignet. Dies liegt unter anderem daran, dass im Ubicompbereich andere Anforderungen an den Entwicklungsprozess und an die resultierende Software gestellt werden. So sind die meisten Applikationen für Netzwerkarchitekturen wie etwa drahtlose Sensornetzwerke(WSNs) ausgelegt, für die es in den herkömmlichen Modellierungssprachen für PIMs keine ausreichenden Mittel gibt. Auch sind die Zieltechnologien wie CORBA im Ubicompbereich nur begrenzt nützlich, da durch die verwendeten Netzwerke und relativ leistungsschwacher Ausführungsplattformen neue Probleme und Begrenzungen entstehen, die

beim Entwurf der für Server und Desktopmaschinen entwickelten Technologien nicht vorgesehen waren. Daher werden im folgenden zwei direkt auf die Entwicklung von ubiquitären Systemen zugeschnittene MDA-Ansätze präsentiert.

2.3 PervML

PervML[2,3,4,5] ist eine Umsetzung der modellgetriebenen Entwicklung für ubiquitäre Systeme. Es implementiert eine eigene Version des MDA-Ansatzes mit eigenem PIM und PSM (vgl. Abb. 2). Dabei nutzt es PervML als Modellierungssprache für seine PIM, daraus wird mittels Transformation ein PSM in einem Metamodell basierend auf OSGi[6][7], einer SOA-Middleware in Java, und schlussendlich wird Javacode erzeugt. PervML wird seit 2004 an der Technical University of Valencia entwickelt. PervML besteht aus der PervML Modellie-

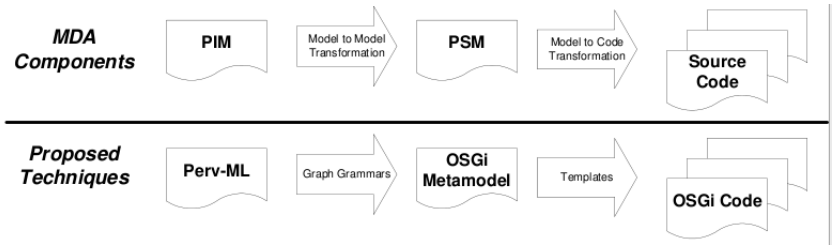


Abb. 2. PervML MDA Umsetzung

rungssprache, einem OSGi Metamodel zur Beschreibung des PSM sowie Transformationen von PervML zu dem OSGi-PSM und von OSGi zu Javacode. Die Modellierungssprache PervML besteht aus zwei Teilen, dem Analystview und dem Architectview (vgl. Abb. 3). Der Analystview dient der Spezifikation der Anforderungen an das System und der prinzipiellen Funktionalität des Systems. Der Architectview nimmt die Spezifikation des Analystview und definiert, wie die abstrakten Services in Hardware und Software umzusetzen sind. Der Analystview besteht aus dem Services Model, dem Structural Model und dem Interaction Model. Das Services Model legt fest, was für Services in dem System zur Verfügung stehen und in welchem Verhältnis diese zueinander stehen. Zu jedem Service werden das Interface und eventuelle Anmerkungen wie Pre- und Postconditions für die einzelnen Operationen spezifiziert, wozu ein erweitertes UML Klassendiagram verwendet wird. Ebenso wird ein UML State Transition Diagram zur Spezifikation des Verhaltens verwendet. Danach wird das Structural Model verwendet um die Umsetzung der einzelnen Services festzulegen indem für jeden Service eine Komponente bereitgestellt wird. Jede Komponente stellt dabei eine Abstraktion der jeweiligen implementierenden Funktionen dar. Das Structural Model wird durch ein UML Component Diagram dargestellt. Am Ende werden durch das Interaction Model das Verhalten die Interaktionen zwischen Services spezifiziert. Dazu gibt es für jede Interaktion ein eigenes UML

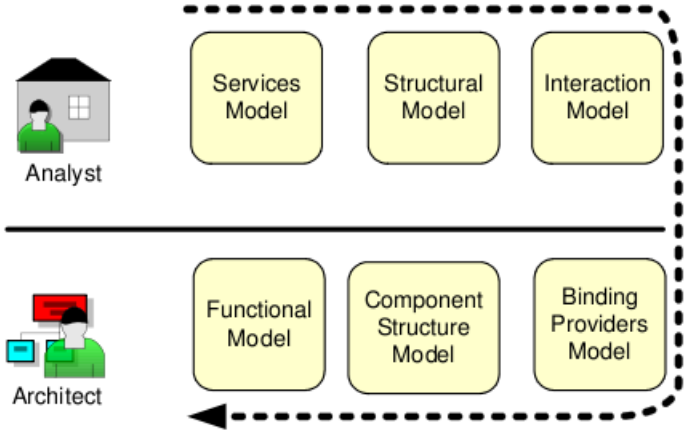


Abb. 3. PervML Aufteilung

Sequenzdiagram indem der Ablauf sowie die auslösende Kondition der Interaktion (festgelegt über Propertywerte) enthalten sind. Nachdem der Analystview abgeschlossen wurde wird der Architectview benutzt. Dieser besteht aus einem Binding Providers Model, einem Component Structure Model und der Component Functional Specification.
Zur Spezifikation des Systems ist (auch) eine Liste der umsetzenden Artefakte notwendig. Dies können Sensoren, externe Softwaresysteme oder andere Geräte sein. Jedes einzelne Artefakt wird dabei ins System als sog. Binding Provider eingebunden. Die verschiedenen Typen dieser Binding Providers sind im Binding Providers Model gelistet. Das Binding Providers Model wird durch ein stereotypiertes UML Klassendiagram definiert. Danach muss jede der Komponenten aus dem Structure Model eine Umsetzung durch Binding Providers erfolgen. Dies geschieht im Component Structure Model, wo jeder Komponente eine Liste von Instanzen von Binding Providers zugeordnet wird. Schlussendlich wird die Component Functional Specification verwendet um die Logik der einzelnen Komponenten festzulegen. Zu diesem Zweck wird für jede Operation eine Sequenz von Aktionen (z. B. Aufrufe der einzelnen Binding Providers) in UML Action Semantic Language festgelegt.
Nachdem die Spezifikation in PervML abgeschlossen ist, wird die PIM mittels Graphtransformation in eine PSM transformiert. Dazu wird die Spezifikation in einem Graphen verwandelt, auf den dann Regeln wie in Abb. 4 angewandt werden. Nachdem diese Transformation geschehen ist erhält man ein PSM im OSGi-Metamodel. Daraus kann mithilfe von Templates für die einzelnen Modellelemente Javacode erzeugt werden, der dann auf die einzelnen Teile des Systems verteilt wird um das System aufzusetzen.
Zum Arbeiten mit PervML steht eine Integration in die Eclipse-Entwicklungsumgebung[8] zur Verfügung. Das PervML Generative Tool (PervGT) stellt graphische Modellierungswerkzeuge für alle Teile von PervML bereit. Darüber hinaus

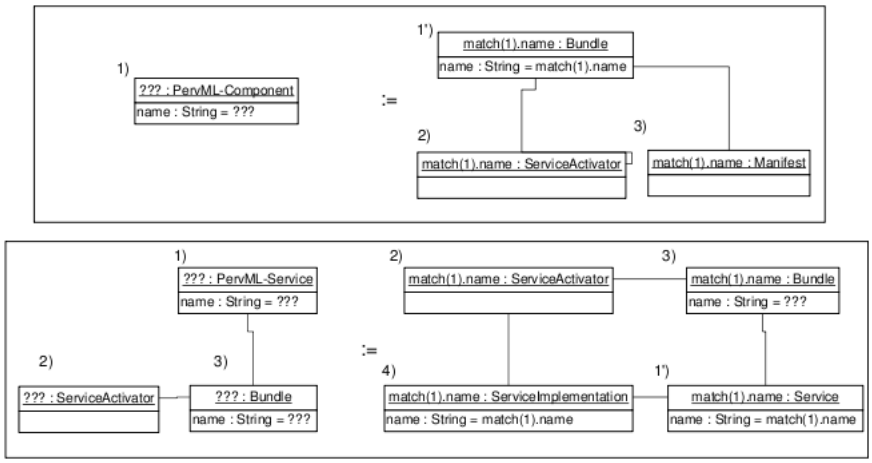


Abb. 4. Graphtransformationsregel

erlaubt PervGT die automatische Transformation des Modells in Javacode, der dann auch weiter in Eclipse entwickelt werden kann.

2.4 Visual Robotics Development Kit (VRDK)

VRDK[9][10] ist ein System zur graphischen Modellierung und Entwicklung von ubiquitären Anwendungen. Es stellt eine graphische Modellierungssprache zum Spezifizieren eines PIM bereit. Die so entworfenen Programme können dann in verschiedene Sprachen übersetzt werden (Zurzeit stehen als Zielsprachen C# und C zur Verfügung, dies kann jedoch durch Plugins leicht erweitert werden). VRDK wird seit 2005 an der TU Berlin entwickelt und ist in C# implementiert. Das Kernstück des VRDK ist eine graphische Sprache zur Modellierung von ubiquitären Applikationen. Dadurch dass das Programm als Modell entsteht und erst später übersetzt wird kann eine homogene Programmierung vieler unterschiedlicher Geräte erreicht werden. Zum Einbinden neuer Geräte wird einfach ein neues Plugin (als .NET Bibliothek(DLL)) hinzugefügt. Darüber hinaus kann jedes Geräteplugin die Sprache um neue Befehle und Events zum Ansteuern der Geräte erweitern. Die im Beispiel (Abb. 5) verwendeten Befehle zum Bedienen eines DVD-Players und einer Lampe sind solche Befehle. Die Sprache selbst bietet Befehle zur Flusskontrolle wie Verzweigungen, Schleifen und parallele Ausführung sowie zum Auswerten einfacher Formeln. Ein Prozess entspricht also im Wesentlichen einem imperativen Programm, welches auf dem ausgewählten Gerät läuft und auf dessen Funktionen zugreift. Darüber hinaus kann ein Prozess mit *wait* auf bestimmte Nachrichten warten oder auch mit *select* abhängig von der einkommenden Nachricht einen Zweig ausführen. Diese Nachrichten können entweder vom Gerät ausgelöst werden oder von anderen Prozessen verschickt werden. Verschiedene Prozesse laufen dabei unabhängig

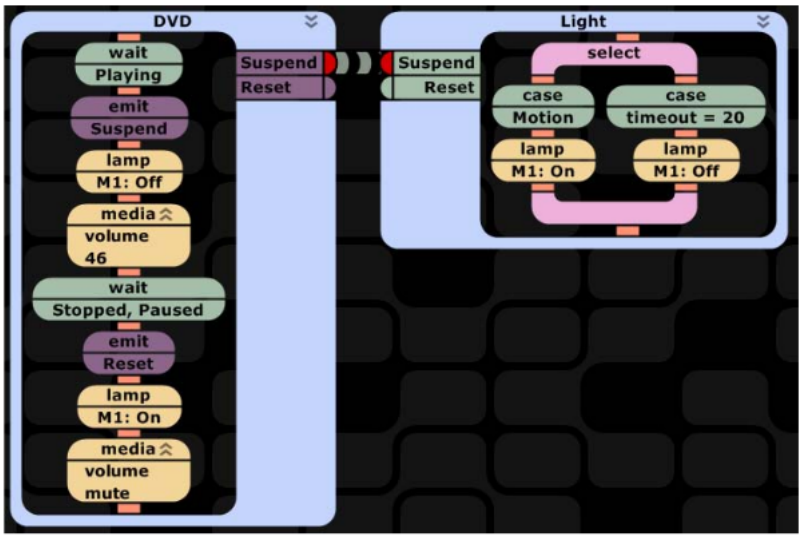


Abb. 5. VRDK Programm

voneinander und möglicherweise auch auf verschiedenen Geräten und können außer über Nachrichten sich nicht gegenseitig beeinflussen. So sorgt im Beispielprogramm der DVD-Prozess dafür, dass der Licht-Prozess anhält, wenn eine DVD spielt, indem er eine *Suspend*-Nachricht an den Prozess sendet. Daneben kann über ein Plugin ein Lokationssystem mit in die Entwicklung einbezogen werden. Dann lassen sich einzelne Prozesse neben Geräten auch Lokationen wie etwa einem Raum zuordnen (vgl. Abb. 7). Wenn das Programm dann später verteilt wird erhalten alle passenden Geräte im Raum den Prozess. Wenn also mehrere Lichter im Raum sind wird der Lichtprozess an alle verteilt. Außerdem können in den Prozessen auch Anfragen an das Lokationssystem gestellt werden um z. B. alle Lichter in einem Raum zu erhalten.
Nachdem das Programm mit der graphischen Sprache erstellt wurde, also ein PIM des gewünschten Systems erstellt wurde, kann der Compiler die passende PSM für jedes Gerät erstellen woraus dann der Code gewonnen wird. Zielplattformen lassen sich wie gehabt per Plugin hinzufügen. Ein besonderes Feature des VRDK ist der Interpreter für das erstellte PIM. Durch Interpretation des Programms auf dem Entwicklungsrechner kann das Programm testweise ausgeführt werden. Dabei werden die einzelnen beteiligten Geräte vom Rechner ferngesteuert und alle Events erst zum PC geschickt. Dies erlaubt Fehlersuche und -Beseitigung auf dem PIM-Niveau ohne zuvor eine PSM zu generieren. Dies steht im Kontrast zu üblichen MDA-Systemen wo die Ausführung erst nach Generierung der PSMs geschieht und bei etwaigen Fehlern Fehler im PIM von Fehlern im generierten PSM unterschieden werden müssen was das Debuggen besonders bei verteilten Applikationen schwierig macht.

Abb. 6. Liste von verfügbaren Geräten

2.5 Fazit

Beide hier vorgestellten Ansätze sind speziell auf ubiquitäre Anwendungen zugeschnitten und obwohl nur in prototypischer Implementierung verfügbar durchaus für Entwicklung größerer Applikationen geeignet. Durch die Abstraktion durch das PIM eignen sich die MDA-Ansätze gut für heterogene Systeme. Auch erleichtert dies die Portierung einer Anwendung auf neue Systeme, dies erfordert nur eine (automatisierte) Transformation von PIM zum passenden PSM. Desweiteren erlaubt dies dem Entwickler, sich auf die eigentliche Applikation zu konzentrieren. So wird der Komplexitätsgrad möglichst gering gehalten und auch komplexe Applikationen sind dann relativ einfach zu entwerfen. Bei der Programmierung muss jedoch das individuelle Verhalten jedes Gerätes genau spezifiziert werden. Dies erschwert die Programmierung drahtloser Sensornetzwerke, die aus einer großen Anzahl von Sensorknoten bestehen. Hier will man eigentlich eher das globale Verhalten als die einzelnen Interaktionen der Sensorknoten festlegen. Auch sind die angebotenen Zielsprachen für Geräte mit geringer Leistung nur bedingt geeignet. Um diese Lücke zu schließen gibt es spezielle Makroprogrammierungssysteme für Sensornetzwerke.

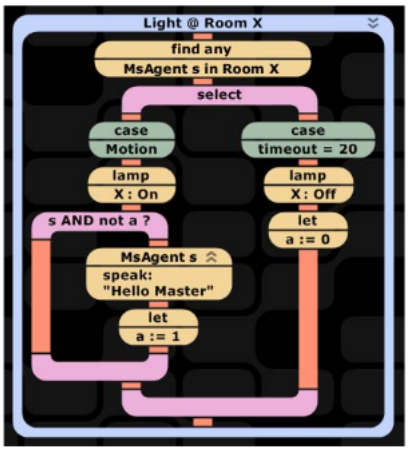


Abb. 7. Lokationsmarkierung der Prozesse

3 Makroprogrammierungssysteme

3.1 Einleitung

Makroprogrammierungssysteme für Sensornetzwerke sollen den Entwicklungsaufwand für die Programmierung von Sensornetzwerken verringern. Dazu wird statt dem im Ubicomp-Bereich gewohnten Bottom-Up Ansatz ein Top-Down Ansatz verfolgt. Dies kann auf verschiedene Arten geschehen, folglich sind viele verschiedene Ansätze publiziert worden. Im Folgenden werden deshalb die Wesentlichen vorgestellt. Dabei ist zu beachten dass diese Arbeiten Prototypen sind, die im Allgemeinen keine graphische IDE zu Verfügung stellen. Auch wurde keines der Systeme in irgendwelchen Anwendungen (außer der jeweiligen Evaluation) eingesetzt.

3.2 RuleCaster

RuleCaster[11][12][13] ist ein regelbasiertes Makroprogrammierungssystem. Es besteht aus einem Compiler für PCs und einer Laufzeitumgebung für TinyOS[14]. RuleCaster wird seit 2005 an der Lancaster University entwickelt und wurde seitdem schon einmal überarbeitet, was sich unter anderem in veränderter Syntax niederschlägt.

In Abb. 8 ist ein Beispielprogramm geschrieben in RCAL, der Programmiersprache für das RuleCaster-System. Das Beispielprogramm soll den Bewohner warnen, falls der Herd in der Küche noch an ist, wenn man das Haus verlässt. Dazu werden zwei Bereiche definiert, *SPACE(kitchen)* die Küche mit Sensoren für den Herd (*SENSOR(stoveOn/0)*,*SENSOR(stoveOff/0)*) und einem Bewegungssensor *SENSOR(motion/1)*, sowie der Eingangsbereich *SPACE(corridor)* mit einem Alarm *ACTUATOR(alarm/1)*. Im Küchenbereich gibt es zwei Regeln. Die

```
SPACE(kitchen) {
INTERFACE:
SENSOR(stoveOn/0),
SENSOR(stoveOff/0),
SENSOR(motion/1),
STATE(stoveOnHazard).

PRE_STATE() : time(2) [
STATE :- stoveOn(), motion(X), X=0.
] POST_STATES(stoveOnHazard).


PRE_STATE(stoveOnHazard) [
STATE :- stoveOff().
     STATE :- motion(X), X>=4.
] POST_STATES().
}
SPACE(corridor) {
INTERFACE:
SENSOR(mail/0),
        ACTUATOR(alarm/1).

    PRE_STATE() : STATE(kitchen:stoveOnHazard) [
        STATE :- alarm(10).
    ] POST_STATES(alarmOn).

    PRE_STATE(alarmOn) : STATE(NOT kitchen:stoveOnHazard) [
        STATE :- alarm(0).
    ] POST_STATES().
}
```

Abb. 8. RuleCaster Programm zum Überwachen eines Küchenherds

erste Regel wird periodisch ausgeführt und hat keine Zustände von denen die Ausführung abhängt (*PRE_STATE() : time(2)*). Falls der Herd an ist und keine Bewegung festgestellt wird so wird der Zustand *stoveOnHazard* angenommen. Die zweite Regel wird ausgeführt falls *stoveOnHazard* besteht und setzt den Zustand zurück falls der Herd aus ist oder der Bewegungssensor etwas registriert. Im Bereich *SPACE(corridor)* wird überprüft ob *STATE(kitchen:stoveOnHazard)* gilt und dann der Alarm aktiviert. Genauso wird der Alarm deaktiviert, falls *STATE(kitchen:stoveOnHazard)* nicht mehr gilt. Zu beachten ist dass sich dieses Programm nicht auf dem tatsächlichen Aufbau des Netzwerks agiert. Stattdessen stellen die *INTERFACE*-Definitionen Anforderungen an das zu benutzende Netzwerk dar. Daher wird noch ein Netzwerkmodell benötigt bevor man das Makroprogramm auf ein Sensornetzwerk umsetzen kann. In RuleCaster gibt es dazu auf jedem Sensorknoten Informationen über die *SPACE*-Zugehörigkeit und den auf dem Sensorknoten verfügbaren Funktionen. Der RuleCaster-Compiler nimmt ein RCAL-Programm und ein Netzwerkmodell und generiert daraus den Bytecode für jeden einzelnen Sensorknoten. Zuerst wird wie gewohnt das Programm eingelesen und für jede Regel ein Ausführungsgraph generiert. Danach kommt der eigentlich wesentliche Teil in dem das Programm auf das Netzwerk angepasst wird. Die Netzwerkbeschreibung wird ausgelesen und überprüft ob das gegebene Netzwerk die Anforderungen erfüllt. Falls dem so ist, werden zuerst Aufrufe an Sensoren und Aktoren an bestimmte Sensorknoten gebunden. Dabei werden alle Sensoren in dem Bereich, die die gewünschte Funktion erfüllen, mit einbezogen. So wird ein Aufruf

$$STATE :- motion(X), X >= 4$$

erfüllt, indem alle Bewegungssensoren überprüft werden. Nachdem diese Zuordnung beendet ist, werden noch die allgemeinen Befehle verteilt. Dies beinhaltet

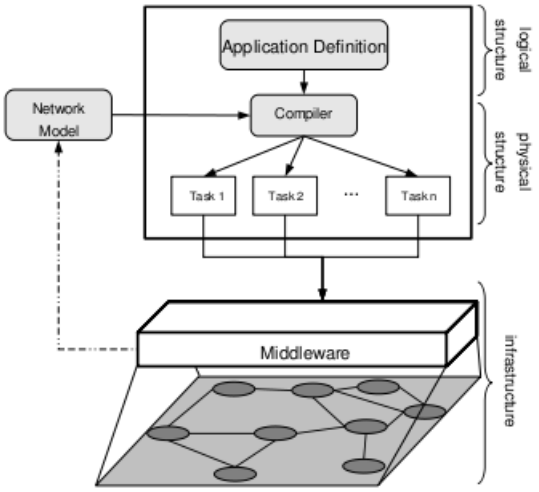


Abb. 9. Aufbau des RuleCastersystems

das Setzen neuer Zustände, arithmetische Operationen etc. Der Programmfluss im verteilten System fließt dann jeweils an den verteilten Befehlen entlang. Am Ende wird dann simpler Bytecode für die RuleCaster-Laufzeitumgebung generiert, welche für die Ausführung der Programme verantwortlich ist.

```
1 iload 0 0 //push data field 0 from channel 0 onto stack
2 ipush 20 //push the value 20 onto the stack
3 eval lessThan //call the service lessThan
4 ifeq 2//jump over next line if service evaluation has returned 0
5 send 0 //send data in channel 0
6 end //operator execution finished
```

Abb. 10. Generiertes Programm für die Rulecaster-Laufzeitumgebung

Sie führt ein lokales Programm wie in Abb. 10 aus. Durch die bewusst simple Laufzeitumgebung läuft Rulecaster auch auf eher billigen Sensoren. Ebenso wird die unterliegende Hardware abstrahiert so dass auch eine Mischung von verschiedenen Sensoren verwendet werden kann.

3.3 MagnetOS

MagnetOS[15][16] ist ein verteiltes Betriebssystem für AdHoc-Sensornetzwerke. MagnetOS führt Javaprogramme aus indem es als verteilte JavaVM agiert, welche statisch und dynamisch Code verteilen kann. Dies führt zu im Vergleich zu anderen Ansätzen gestiegene Anforderungen an die Sensorknoten. Es wird seit 2001 an der Cornell University entwickelt. Ein MagnetOS Programm besteht aus

einer Menge verteilter Eventhandler, die jeweils von der MagnetOS Runtime verwaltet werden. Der Input für MagnetOS ist ein normales Javaprogramm, welches mittels einer API auf die einzelnen Funktionen der Sensoren zugreifen kann. Ein spezieller Compiler teilt dieses in einzelne Klassen auf und fügt diesen einen Eventhandler hinzu. Darüber hinaus werden Zugriffe auf andere Klassen sowie die Instantiierung neuer Objekte durch Aufrufe an die Runtime ersetzt. Die einzelnen Teile werden dann nach statischer Analyse über das Netzwerk verteilt, wo das Programm dann durch Aufruf des ersten Eventhandlers gestartet wird. Mit jedem neuen Objekt wird auch ein neuer Eventhandler registriert. Aufrufe von anderen Klassen werden durch die Erstellung und Versendung eines passenden Events erreicht. Während der Laufzeit laufen mehrere Algorithmen ab, welche dynamisch die Codeverteilung zu optimieren versuchen. Dazu werden eingeteilt in diskrete Zeitabschnitten verschiedene Parameter des Netzwerks gemessen. Ein Algorithmus misst z. B. das Kommunikationsvolumen zwischen direkten Nachbarn und verschiebt eine Klasse in Richtung der größten Kommunikation, ein anderer versucht zusammengehörige Klassen zu Clustern zusammenzufassen. Es ist ebenfalls möglich Code von Knoten mit fast leerer Batterie weg zu bewegen. Falls es zu Netzwerkversagen kommt kann der Programmierer applikationsspezifisch das weitere Verhalten bestimmen, je nachdem ob eine Applikation sich noch regenerieren kann.

3.4 COSMOS

COSMOS[17][18] ist ein Makroprogrammierungssystem für Sensornetzwerke. Ein COSMOS Programm besteht aus einer Kombination verschiedener vorhandener Funktionen. COSMOS wird seit 2007 an der Purdue University entwickelt und es gibt eine Referenzimplementation für Mica2[19] Sensorknoten sowie POSIX Systeme.
COSMOS besteht aus einer Programmiersprache mPL und einem Betriebssystem für Sensorknoten mOS. mPL Programme bestehen aus einem Datenflussgraph durch verschiedene Functional Components (FCs). Eine FC besteht dabei aus einer Interfacedeklaration sowie einem kleinen C Programm. Wenn Daten in eine FC hineinkommen, wird das C Programm ausgeführt und produziert neue Daten, die dem Graph entlang in neue FCs gefüttert werden. Neue FCs können zur Laufzeit an die Sensorknoten verteilt werden. Der mPL Compiler generiert einen annotierten gerichteten Graphen der über das Netzwerk verteilt wird, wobei jeder Sensorknoten nur den Teilgraphen erhält, für den er die nötigen FCs besitzt. Das mOS Betriebssystem ist dann für die Instantiierung der einzelnen FCs zuständig. mOS kümmert sich um die Verwaltung der FCs sowie der Kommunikation über das Netzwerk. Durch den Einsatz eines separaten Betriebssystems kann COSMOS auf heterogenen Sensornetzwerken laufen.

3.5 Regiment

Regiment[20][21][24] ist eine funktionale reaktive Programmiersprache für Sensornetzwerke. Regiment wird seit 2004 am MIT und Harvard entwickelt. Die

```
// declarations (auto import)
%accel_x   : mcap = MCAP_ACCEL_SENSOR,
             device = ACCEL_X_SENSOR, out[raw_t];
%cpress_fc : { mcap = MCAP_ANY, fcid = FCID_CPRESS,
               in[raw_t], out[craw_t] };
%thresh_fc : { mcap = MCAP_ANY, fcid = FCID_THRESH,
               in[craw_t, ctrl_t], out[craw_t] };
%ctrl_fc   : { mcap = MCAP_ANY, fcid = FCID_CTRL,
               in[max_t], out[ctrl_t] };
%max_fc    : { mcap = MCAP_ANY, fcid = FCID_MAX,
               in[craw_t, max_t], out[max_t]};
%disp      : mcap = MCAP_UNIQUE_SERVER,
             device = DISPLAY, in[ * ];
%fft_fc    : { mcap = MCAP_FAST_CPU, fcid = FCID_FFT,
               in[craw_t], out[freq_t] };
// logical instances
accel_x   : accel(12);
disp      : disp1, disp2;
cpress_fc : cpress;
thresh_fc : thresh(250);
max_fc    : max;
fft_fc    : fft;
ctrl_fc   : ctrl;

// refining capability constraints
@ on_mote = MCAP_ACCEL_SENSOR : thresh, cpress;
@ on_srv = MCAP_UNIQUE_SERVER : ctrl;

start_ia
timer(30) -> accel;
accel     -> cpress[0];
cpress[0] -> thresh[0], max[0];
thresh[0] -> fft[0];
fft[0]    -> disp1;
max[0]    -> ctrl[0], disp2 | max[1];
ctrl[0] -->  thresh[1];
end_ia
```

Abb. 11. mPL Programm zur Verarbeitung und Darstellung von Beschleunigungsmessungen

Referenzimplementation generiert Code für eine eigene auf TinyOS laufende Ausführungsumgebung.

```
dosum :: float, (float, int) -> (float, int)
fun dosum(temp, (sumtemp, count)) {
   (sumtemp+temp, count+1)
}
tempreg = rmap(fun(nd){sense("temp",nd)}, world);
sumsig = rfold(dosum, (0,0), tempreg);
avgsig = smap(fun((sum,cnt)) {sum / cnt},
               sumsig);
BASE <- avgsig
```

Abb. 12. Regimentprogramm zum Ermitteln der Durchschnittstemperatur

Regiment behandelt Sensormessungen als Streams von Signalen indem es einen Signal-Datentyp bereitstellt. Indem man Nodes gruppiert kann man mehrere Signale als abstrakte Regionen manipulieren. Dazu stehen die Standardmittel der funktionalen Programmierung fold,map und filter bereit. Eine abstrakte Region muss dabei nicht unbedingt aus räumlich nahen Knoten bestehen und kann sich

auch während der Laufzeit ändern. Ein Regimentprogramm besteht dann aus einem funktionalen Programm mit diesen Primitiven sowie einer Zuweisung

$$BASE \leftarrow Wert$$

welche den an die Basisstation übermittelten Wert bestimmt. Der Regiment Compiler geht in mehreren Schritten vor und generiert ein event-gesteurtes Programm für jeden Knoten. Dabei wird zuerst das Programm durch mehrere Reduktionen normalisiert. Danach werden Regionen in lokale Streams mit spezifischen IDs umgewandelt und ein lokaler Datenflussgraph für jeden Sensorknoten erstellt. Das entstehende lokale Programm läuft auf einer Tokenmaschine[22], eine Ausführungsumgebung auf TinyOS. Die Tokenmaschine bietet eine simple Programmiersprache zum Schreiben von verteilten Programmen auf Basis von Behandlung und Verteilung von sog. Tokens (typisierte Daten die beim Empfang ein Event auslösen). Tokens erlauben eine simple Behandlung von Verteilung und Speicherung von Variablen. Ein Programm für die Tokenmaschine besteht aus verschiedenen Eventhandlern für den Empfang von bestimmte Tokens wobei jeder Handler neue Tokens versenden kann. Durch spezielle Tokens zur Erstellung von und Zugehörigkeit zu Regionen kann Regiment auf relativ simples Verhalten auf dem Knotenniveau übersetzt werden.

Es existiert eine Weiterentwicklung von Regiment namens Wavescript[23] und läuft auf Wavescope, einem ebenfalls am MIT entwickelten System zum Verarbeiten von Signalstreams. Ein Vorteil der neuen Version ist dass sie direkt C Code für linuxbasierte Geräte generiert. Darüber hinaus bietet es weitere Features zum Schreiben von verteilen Applikationen auf Netzwerken.

3.6 Kairos

Kairos[25][26][27] erweitert eine Programmiersprache um Primitive zur Manipulation von Sensorknoten. Ein mit diesen Primitiven geschriebenes Programm kann dann in eine verteilte Applikation transformiert werden. Kairos wird seit 2005 vom Embedded Networks Laboratory an der Universität von Süd Kalifornien entwickelt. Die Referenzimplementation ist für Python und läuft auf Stargates[19] und Mica2Dots.
Kairos fügt der Programmiersprache 3 Primitive hinzu:

- Sensorknoten als Sprachprimitive, zusätzlich iterierbare Listen von Sensorknoten, vgl. Abb.13 Z. 2+4
- Zugriff auf lokale Nachbarn eines Knotens, vgl. Abb.13 Z. 12
- Zugriff auf die Variablen eines bekannten Sensorknotens, es funktioniert wie eine Art gemeinsamer Speicher für alle Knoten, vgl. Abb.13 Z. 18+19, var@node

Dies erlaubt es ein sequentielles Programm für das Sensornetzwerk zu schreiben welches dann in einzelne Programme für die spezifischen Knoten transformiert wird. Der so entstandene Code kann dann vom normalen Compiler kompiliert werden. Dazu stellt Kairos eine Runtime und einen Präprozessor bereit, der die

```
1:  void buildtree(node root)
2:    node parent, self;
3:    unsigned short dist_from_root;
4:    node_list neighboring_nodes, full_node_set;
5:    unsigned int sleep_interval=1000;
      //Initialization
6:    full_node_set=get_available_nodes();
7:    for (node temp=get_first(full_node_set); temp!=NULL;
           temp=get_next(full_node_set))
8:      self=get_local_node_id();
9:      if (temp==root)
10:       dist_from_root=0; parent=self;
11:     else dist_from_root=INF;
12:     neighboring_nodes=create_node_list(get_neighbors(temp));
13:   full_node_set=get_available_nodes();
14:   for (node iter1=get_first(full_node_set); iter1!=NULL;
           iter1=get_next(full_node_set))
15:     for(;;) //Event Loop
16:       sleep(sleep_interval);
17:       for (node iter2=get_first(neighboring_nodes);
                iter2!=NULL; iter2=get_next(neighboring_nodes))
18:         if (dist_from_root@iter2+1<dist_from_root)
19:           dist_from_root=dist_from_root@iter2+1;
20:           parent=iter2;
```

Abb. 13. Kairosprogramm zum Erstellen eines Shortest Path Routing Tree

Primitive in Aufrufe der Runtime übersetzt. Das Produkt des Präprozessors ist dann in komplett in der Originalsprache. Zu beachten ist dass auf jedem Knoten dasselbe Programm abläuft. Die Runtime kümmert sich um die Kommunikation zwischen und um die Bereitstellung von extern liegenden Daten. Die Daten werden lose synchronisiert, dazu befindet sich auf jedem Knoten ein Cache für Variablen, der sich wie gewohnt verhält. Dies reduziert den benötigten Kommunikationsoverhead.

In einem weiteren Paper[28] stellen die gleichen Forscher eine Erweiterung für Makroprogrammierungsysteme vor, die das Abfangen und Behandeln von Fehlern mittels eines simplen Checkpointsystems ermöglichen soll. Diese Erweiterung wurde für Kairos implementiert und getestet.

Als Weiterentwicklung von Kairos wurde Pleinades[29] vorgestellt. Es basiert auf denselben Ideen wie Kairos, ist jedoch für C implementiert und produziert nesC Code. Außerdem enthält es fortgeschrittenere Programmzerteilungs- und verteilungsmechanismen sowie Unterstützung für gleichzeitige Ausführung von verteilten Programmteilen (dies ist wichtig für die Beibehaltung der Synchronisation).

3.7 ATaG

ATaG (Abstract Task Graph)[30] [31] [32] [33] ist ein Makroprogrammierungsframework für WSNs welches auf einer sog. datengetriebenen Programmierung aufbaut. Es wird seit 2005 an der Universität von Süd Kalifornien entwickelt. Es existiert eine Referenzimplementation mit imperativen Code in Java und deklarativen Teil in XML, desweiteren wird das Generic Modeling Environment[34], einem erweiterbaren Tool zur graphischen Modellierung, eingesetzt.
Ein ATaG Programm besteht im Wesentlichen aus Javaprogrammen, dazu Re-

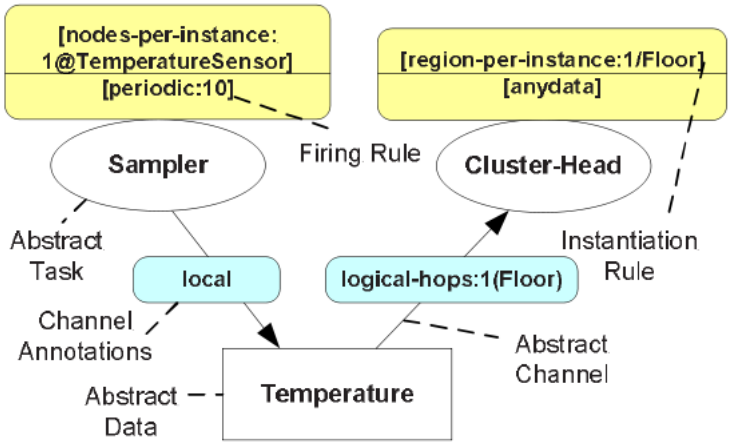


Abb. 14. ATaG Programm zur Bestimmung der Temperatur

geln wie in Abb. 14, die die Verteilung und Ausführung der Programme bestimmen sowie Kanälen zwischen den Programmen. Formell setzt es sich aus einer Reihe von abstrakten Tasks, im System vorkommenden Datentypen, sowie Verbindungen zwischen Tasks und Datentypen zusammen. Zusammengesetzt bestimmen die Verbindungen zwischen Daten und den verarbeitenden Tasks die globale Logik indem sie den Datenfluss vorgeben. Jeder Task verarbeitet Daten aus einkommenden und produziert Daten für ausgehende Verbindungen. Die Tasks setzen sich dabei aus einem deklarativen und imperativen Teil zusammen. Diese Aufteilung ist es, die bei diesem Framework die Makroprogrammierung ermöglicht. Der imperative Teil ist ein Programm welches wie gewohnt auf einen Sensorknoten zugeschnitten ist und je nach Richtung der Verbindung Daten annimmt oder produziert, das Interface beschränkt sich daher auf putData() und getData(). So könnte ein typisches Programm einen Sensor auslesen oder einen Mittelwert über einkommende Daten bilden. Der deklarative Teil besteht aus einer Liste von Regeln die die Ausführung des imperativen Teils kontrollieren. Solche Regeln bestimmen u.a. wo der Task instantiiert wird (dies kann eine Regel wie einmal alle 4 Knoten, nur auf einem bestimmten Knoten oder auch einer in jedem Raum sein) und wann der Task ausgeführt wird (periodisch, beim Empfangen von Daten etc.). Die Verbindungen zwischen Tasks können ebenfalls bestimmte Regeln besitzen die bestimmen welche Daten gesammelt werden (lokal, alle Daten aus einem Umkreis von x Metern etc.) oder wie die Daten bereit gestellt werden.
Durch einen Compiler werden die Tasks und Verbindungen auf ein Sensornetzwerk gemappt indem eine Konfiguration für die benötigten Sensorknoten (bestehend aus Tasks, deren Aktivierungsregeln sowie eine Liste von Konsumenten für die produzierten Daten) generiert wird. Dabei erfordern manche Regeln (z. B. lokationsabhängige Angaben) Unterstützung im Compiler und der

Ausführungsumgebung. Genauso können einfach neue Regeltypen hinzugefügt werden indem man den Compiler erweitert. Der Compiler übernimmt außerdem das Verteilen des Codes auf die individuellen Knoten, wo eine spezielle Ausführungumgebung genannt DART (Datadriven ATaG RunTime) sich um die Ausführung der einzelnen Tasks kümmert.

4 Zusammenfassung und Kritik

Wie gesehen gibt es für den Ubicomp Bereich verschiedene Umsetzungen der modellgetriebenen Entwicklung, welche auch durchaus einen ausgereiften Eindruck machen. Sie erlauben die Entwicklung komplexer Systeme und unterstützen dabei den Entwickler mithilfe von graphischen Modellierungswerkzeugen, was zum produktiven Arbeiten stark beiträgt. PervML ist eine Umsetzung der MDA-Spezifikation für ubiquitäre Applikationen. Dies wird erreicht indem eine neue Spezifikationssprache namens PervML verwendet wird, die speziell auf ubiquitäre Applikationen zugeschnitten ist. Aus dem so erstellten PIM kann dann durch Modelltransformation ein OSGi-PSM erstellt werden, aus dem sich direkt Javacode generieren lässt. Das zweite vorgestellte System ist das VRDK, welches eine graphische Modellierungssprache und Oberfläche bietet. Hier ist besonders zu bemerken, dass es leicht möglich ist dem System durch Plugins neue Gerätetypen hinzuzufügen. Diese Plugins sind für die Ansteuerung und spätere Codegenerierung verantwortlich und werden genauso wie das VRDK selbst in .NET geschrieben. Dadurch bleibt das Basissystem auf die nötigsten Befehle beschränkt und es kann eine diverse Mischung von Geräten und den dazu gehörenden Programmiersprachen verwendet werden. Auch erlaubt ein eingebauter Interpreter die direkte Evaluierung des PIM, was bei der Fehlersuche und -behebung hilft. Beide Systeme bieten eine graphische Oberfläche zur Unterstützung des Programmierers, PervML durch Integration in die Entwicklungsumgebung Eclipse und VRDK durch ein eigenes Programm. Wegen der angebotenen Zielsprachen eignen sie sich besonders für die Programmierung relativ leistungsfähiger Systeme obwohl VRDK durch geschickte Einsetzung des Pluginmechanismus dies zumindest teilweise umgehen kann. Auch erlauben die zur Verfügung stehenden Primitive es nur bedingt eine große Anzahl von Geräten gleichzeitig zu verwenden da jedes Gerät einzeln programmiert werden muss.
Um diese Fehler zu kompensieren wurde eine speziell hierfür entwickelte Klasse von Systemen vorgestellt: die Makroprogrammierungssysteme. Allen Systemen ist gemein, dass sie ein globales Programm nehmen und dieses je nach gegebenen Netzwerk (welches von System zu System unterschiedlich präzise modelliert wird) auf- und verteilen. Die Unterschiede liegen im Wesentlichen in der verwendeten Sprache und auf den Sensorknoten laufenden Laufzeitumgebung. Diese Unterschiede sind durch die jeweils unterschiedlichen Verwendungszwecke bestimmt. Das erste gezeigte System ist Rulecaster, welches versucht, durch eine Menge von an Lokationen gebundene Zustände den globalen Status zu modellieren. Durch Wenn-Dann Regeln werden Zustandsübergänge ausgelöst, die wie-

derum neue Regelausführungen zur Folge haben können. So kann ein Ereignis in einem Raum dort durch eine Regel festgestellt werden, was dann durch Setzen eines Zustands eine Benachrichtigung in einem anderen Raum auslöst. Rulecaster bietet ein statisches Lokationssystem und erlaubt die Einbindung von Aktoren um die Umgebung zu manipulieren. MagnetOS ist eine verteile JavaVM, die ein gegebenes Javaprogramm im Hinblick auf Performance zerteilt. Ein besonderes Feature ist die Unterstützung von AdHoc-Netzen durch Performancemessungen während der Laufzeit und daraus folgende dynamische Neuverteilung von Programmteilen. Je nach Applikation können auch Verluste von Teilen des Netzwerks und die daraus entstehenden Daten- und Programmteilverluste repariert werden. Die Verwendung von Java sorgt jedoch dafür dass die Systemanforderungen an die einzelnen Sensorknoten im Vergleich zu den anderen Ansätzen erhöht ist. Das Makroprogrammierungssystem COSMOS fokussiert sich auf das schnelle Erstellen von Programmen mittels Verwendung von vordefinierten Funktionen. Ein COSMOS Programm besteht aus einem Datenflussgraphen durch eine Menge solcher vordefinierter Module. Diese bestehen aus einem C-Programm, welches z.B. einen Sensor auslesen oder einen Aktor auslösen kann, sowie einer Interfacedeklaration und müssen vor der Programmzerteilung auf den einzelnen Sensorknoten vorhanden sein. Der Datenflussgraph wird dann durch den Compiler auf das Netzwerk abgebildet. Regiment setzt auf ein funktional-reaktives Design zur Programmierung eines Sensornetzwerks. Es bietet die üblichen funktionalen Methoden, entsprechend angepasst für Sensorknoten und erlaubt eine gesonderte Behandlung von Mengen von Sensorknoten als abstrakte Regionen. Regiment eignet sich besonders zur Signalverarbeitung und sieht nicht vor Aktoren einzubinden. Ein Ansatz eine bestehende Sprache durch möglichst geringe, aber sinnvolle Erweiterungen für Sensornetzwerke anzupassen ist Kairos. Kairos fügt Python einige Konstrukte zur direkten Manipulation von Sensorknoten hinzu. Durch Einsetzen eines Präprozessors werden die Konstrukte dann in Aufrufe der Kairoslaufzeitumgebung umgesetzt. Dies erlaubt es dann den Originalcompiler zu verwenden. Ein Nachteil dabei jedoch ist dass keine besondere Zerteilung stattfindet und daher auf jeden Sensorknoten dasselbe Programm gelangt. Außerdem ist es nicht vorgesehen gezielt Sensormessungen auszulösen oder Aktoren auszulesen. Das letzte vorgestellte Makroprogrammierungssystem ist ATaG, wobei einzelne Javaprogramme mit Annotationen versehen werden. Diese Annotationen bestimmen wo und wann die Programme ausgeführt werden und von welchen zu welchen Knoten die Daten fließen. Da die Annotationen erst vom Compiler ausgewertet werden und außer einer festen Syntax beliebige Regeln enthalten können, kann durch simple Erweiterung des Compilers z.B. ein neues Lokationssystem eingebunden werden. Allen vorgestellten Makroprogrammierungssystemen ist gemein, dass sie mit Hinblick auf eine fixe Idee entwickelt wurden und daher relativ spezifisch sind. Von allen Systemen ist wahrscheinlich Rulecaster am ehesten in bestehende MDA-Ansätze einzubinden, da es die Programme relativ abstrakt sind und es durch das Lokationssystem und Einbindung von Aktoren zu den Mächtigsten von allen Ansätzen zählt.
Daraus folgt, dass das am Anfang der Arbeit als wünschenswert vorgestellte

System so noch nicht existiert. Dabei lässt sich aus der Fülle der bestehenden Systeme schließen, dass das erst seit einigen Jahren bestehende (alle Systeme sind ab 2004) Gebiet der modellgetriebenen Entwicklung für ubiquitäre Systeme durchaus voranschreitet. Dies wiederum begünstigt die übrige Ubicomp-Forschung, da die Entwicklung von ubiquitären Applikationen von plattformunabhängiger Modellierung stark profitiert. Anstatt jedesmal die für die aktuelle Forschung unnötigen Details neu implementieren zu müssen kann durch abstrakte Beschreibung und späterer Generation eine Idee schnell auf einer Vielzahl von Geräten ausprobiert werden. Durch Makroprogrammierung entfällt die Bindung an ein bestimmtes Netzwerk und Notwendigkeit zur Umsetzung der meist fehlerträchtigen Koordination zwischen einzelnen Teilen des Systems. Auch Firmen könnten hiervon Gewinn machen. Verwendung von Konzepten der modellgetriebenen Entwicklung erlaubt die Einbindung in schon bestehende Entwicklungsprozesse. Durch geschickte Wahl der Modellierungswerkzeuge kann das benötige Vorwissen über den Ubicomp-Bereich stark reduziert werden, so dass auch nicht speziell ausgebildete Mitarbeiter mit den Systemen umgehen können. Dies erleichtert den Einstieg in die Entwicklung von ubquitären Applikationen, welcher mit der Zunahme des Vernetzungsgrads der vom Endanwender benutzten Geräte jetzt durchaus sinnvoll wird. Dies wird wahrscheinlich eine Vergrößerung des Marktes für ubiquitäre Anwendungen zur Folge haben, wobei dann gute Entwicklungswerkzeuge zum Ausnutzen dieses Effekts notwendig werden.
Aus den oben genannten Gründen und den vorgestellten Systemen lässt sich die Prognose ableiten, dass das gewünschte modellgetriebene System mit Makroprogrammierungsfähigkeiten sowohl wünschenswert als auch machbar ist. Daher ist es zu erwarten, dass in der Folgezeit mehr Systeme publiziert werden, die sich diesem Idealsystem annähern werden und auch dass noch notwendige Konzepte in Beispielsystemen ausgearbeitet werden. Es ist natürlich so, dass eine gezielte Initiative zur Entwicklung eines solchen Systems dies sehr viel schneller hervorbringen wird.

Literatur

1. The official MDA Guide Version 1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf.
2. J. Munoz, 'Pervasive Systems Development with the Model Driven Architecture', 2004
3. J. Muoz, V. Pelechano and J. Fons, Model Driven Development of Pervasive Systems in I International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES 2004), 2004
4. J. Munioz, V. Pelechano and C. Cetina, 'Software Engineering for Pervasive Systems. Applying Models, Frameworks and Transformations' in International Conference on Pervasive Services, July 2007
5. C. Cetina, E. Serral, J. Munoz and V. Pelechano, 'Tool Support for Model Driven Development of Pervasive Systems' in Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software 2007 (MOMPES07), March 2007

6. J. Munoz, V. Pelechano, E. Serral, 'Providing Platforms for Developing Pervasive Systems with MDA. An OSGi Metamodel' in Actas de las Jornadas de Ingeniera de Software y Base de Datos (JISBD), September 2005
7. OSGi Website, http://www.osgi.org/.
8. Eclipse Website, http://www.eclipse.org/.
9. T. Weis, 'Rapid Prototyping of Context-aware Applications' in GI/ITG KuVS Fachgespräche Systemsoftware für Pervasive Computing, 2005
10. A. Ulbrich, T. Weis, K. Geihs, and W. Allee, 'A Modeling Language for Applications in Pervasive Computing Environments', in 2nd Intl. Workshop on Model-Based Methodologies for Pervasive and Embedded Software, 2005.
11. U. Bischoff and G. Kortuem, 'Rulecaster: A macroprogramming system for sensor net- works' in OOPSLA 06 Workshop on Building Software for Sensor Networks, October 2006
12. U. Bischoff, V. Sundramoorthy and G. Kortuem, 'Programming the smart home' in 3rd IET International Conference on Intelligent Environments, 2007
13. U. Bischoff and Gerd Kortuem, 'A Compiler for the Smart Space' in 2nd European Conference on Ambient Intelligence (AmI 2007), November 2007
14. P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill8, M. Welsh, E. Brewer and D. Culler, 'TinyOS: An Operating System for Sensor Networks' in Ambient Intelligence, Springer, 2005
15. H. Liu, T. Roeder, K. Walsh, R. Barr and E. G. Sirer, 'Design and Implementation of a Single System Image Operating System for Ad Hoc Networks' in The International Conference on Mobile Systems, Applications, and Services (Mobisys), June 2005
16. MagnetOS Website, http://www.cs.cornell.edu/People/egs/magnetos/.
17. A. Awan, A. Grama and S. Jagannathan, 'Macroprogramming Heterogeneous Sensor Network Systems Using COSMOS' in Poster Session of the 3rd Symposium on Networked Systems Design and Implementation (NSDI '06), May 2006
18. A. Awan, S. Jagannathan and A. Grama, 'Macroprogramming Heterogeneous Sensor Network Systems Using COSMOS' in The European Conference on Computer Systems (EuroSys '07), March 2007
19. Crossbow (Stargate, Mica2) Website, http://www.xbow.com/Products/wproductsoverview.aspx.
20. R. Newton and M. Welsh, 'Region Streams: Functional Macroprogramming for Sensor Networks' in Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004 (DMSN04), 2004
21. R. Newton, G. Morrisett and M. Welsh, 'The Regiment Macroprogramming System' in Proceedings of the 6th international conference on Information processing in sensor networks (IPSN07), 2007
22. R. Newton, Arvind and M. Welsh, 'Building up to Macroprogramming: An Intermediate Language for Sensor Networks' in Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN05), 2005
23. L. Girod, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Balakrishnan and S. Madden, 'The Case for a Signal-Oriented Data Stream Management System' in Proceedings of CIDR07, 2007
24. Regiment Website, http://regiment.us.
25. R. Gummadi, O. Gnawali and R. Govindan, 'Macro-programming Wireless Sensor Networks using Kairos' in Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS), June 2005
26. R. Gummadi, N. Kothari, R. Govindan and T. Millstein, 'Kairos: a macroprogramming system for wireless sensor networks' in SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles, 2005

27. Kairos Website, http://kairos.usc.edu.
28. R. Gummadi, N. Kothari, T. Millstein and R. Govindan, 'Declarative Failure Recovery for Sensor Networks' in Proceedings of the Sixth International Conference on Aspect-Oriented Software Development (AOSD), March 2007
29. N. Kothari, R. Gummadi, T. Millstein and R. Govindan, 'Reliable and Efficient Programming Abstractions for Wireless Sensor Networks' in Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2007
30. A. Bakshi, V. K. Prasanna, J. Reich and D. Larner, 'The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems' in Workshop on End-to-End, Sense-and-Respond Systems, Applications and Services(EESR 05), June 2005
31. A. Bakshi, A. Pathak and V. K. Prasanna, 'System-level Support for Macroprogramming of Networked Sensing Applications' in International Conference on Pervasive Systems and Computing (PSC 05), June 2005
32. A. Pathak, L. Mottola, A. Bakshi, V. K. Prasanna and G.P. Picco, 'Expressing Sensor Network Interaction Patterns using Data-Driven Macroprogramming' in International Conference on Pervasive Computing and Communications Workshops (PerComW07), March 2007
33. A. Pathak, L. Mottola, A. Bakshi, V. K. Prasanna and G.P. Picco, 'A Compilation Framework for Macroprogramming Networked Sensors' in Distributed Computing in Sensor Systems (DCOSS07) , June 2007
34. The Generic Modeling Environment, http://www.isis.vanderbilt.edu/projects/gme.

Tab. 1. Vergleich der Makroprogrammierungssysteme

Name	Typ	Programmierungssprache	Fokus	Lokationssystem	Aktoren
RuleCaster	Lokationsgebundene Zustände und regelgesteuerte Übergänge	RCAL	Smart Home	Ja	Ja
MagnetOS	Verteilte Java-VM/Betriebssystem	Java	Applikationen in AdHoc-Netzen	Nein	Ja
COSMOS	Datenflussgraph durch funktionale Einheiten	mPl	Wiederverwendung von vorgegebenen Modulen	Nein	Ja
Regiment	Funktional reaktiv	Regiment	Signalverarbeitung	Nein	Nein
Kairos	Erweiterung einer bestehenden imperativen Sprache	Python	Minimal nötige Erweiterung	Nein	Nein
ATaG	Verteilungsregeln für imperative Teilprogramme	Java (Teilprogramme) + XML (Verteilungsregeln)	Konfigurierbare Verteilung	Ja	Ja