Humboldt-Universität zu Berlin

Institut für Informatik

Ringvorlesung „Selbstorganisierende Systeme“

Bericht von Benjamin Daeumlich

Sommersemester 2007

Inhaltsverzeichnis

0 Vorwort

Dieses Dokument ist im Rahmen der Ringvorlesung „Selbstorganisierende Systeme“ an der Humboldt-Universität zu Berlin, Institut für Informatik, im Sommersemester 2007 entstanden. Aufgabe war es, einen Bericht über die gehaltenen Vorträge zu schreiben.

Ich habe mich dazu entschieden, zu jedem einzelnen Vortrag eine Zusammenfassung zu schreiben, welche den Inhalt des Vortrags in Kurzform wiedergeben soll. Es wurde versucht, die einzelnen Vorträge in logische Abschnitte zu unterteilen, wenn möglich entsprechend der Gliederung des jeweiligen Vortrags.
Auf die Vorträge an sich (Vortragsstil usw.) bin ich dabei nicht eingegangen.

1 Funksensornetze - von der Forschung zum realen Einsatz

von Prof. Jochen H. Schiller, Freie Universität Berlin

1.1 Forschungsgebiete

Im Bereich der WSNs (Wireless Sensor Networks) gibt es die verschiedensten Forschungsgebiete. Es wird beispielsweise versucht, die Frage nach den Anwendungsgebieten zu klären. Des Weiteren sollen sich die Fähigkeiten von WSNs verbessern: Sie sollen robuster und langlebiger werden, sie sollen sich selbst verwalten und im Störfall auch selbst reparieren können.
Außerdem wird gerade versucht, die Anwendung, Benutzung und Programmierung der Sensoren zu vereinfachen. Dabei soll dann Hardware und Software zur Verfügung gestellt werden, mit der dann relativ einfach und unabhängig von der Programmiersprache Sensoren in das gewünschte System integriert werden können. Es existieren z.B. schon Integrationen in Micorsoft Visual Studio oder Eclipse.

1.2 Anwendungsgebiete

Die Anwenungsgebiete liegen z.B. in den Bereichen der Überwachung, der Medizin und des „Intelligenten Gebäudes“. Im Folgenden werden einige konkrete Anwenungsbeispiele vorgestellt:

- Scratching-Detection: Die BVG macht jährlich ca. 8 Millionen Euro Verlust durch zerkratzte Scheiben in ihren Verkehrsmitteln. Beim Zerkratzen ändert sich der Brechungsindex der Scheiben und darauf sollen in die Scheiben integrierte Sensoren ansprechen und Aktionen auslösen, wie z.B. eine Kameraaufzeichnung beginnen (in Berlin darf nur auf begründeten Verdacht gefilmt werden) oder eine Warnung über Lautsprecher ausgeben.
- Baustellenüberwachung: Aufgrund von einem erhöhten Diebstahlaufkommen auf Baustellen sollen Sensoren in die Bauzäune integriert werden, die erkennen, wenn jemand über den Zaun klettert. Daraufhin soll das Sicherheitspersonal informiert werden. Es ist allerdings schwierig, den Klettervorgang an sich von z.B. Gegenlehnen oder Gegenstoßen zu unterscheiden, deswegen liegt die Erkennungsrate im Moment nur bei ca. 30%.
- Wassertemperaturmessung in verschiedenen Wassertiefen
- Vogelüberwachung: Um Vögel nicht mehr beim Brüten stören zu müssen, bekommen sie Sensoren ans Gefieder und und können so überwacht werden.
- Messung von Temperaturen in Berggestein, um Erdrutsche vorherzusagen

1.3 Ausblick

Heutzutage sind WSN bereits verfügbar, es existieren Prototypen für Anwendungen und es existieren viele Möglichkeiten zum Einsatz, wie z.B. bei Überwachungen.
In der Zukunft wird sich aber noch einiges ändern. Vor allem die Lebenszeit und Robustheit der Netze soll sich verbessern. Nach Möglichkeit sollen sich WSNs zudem selber Konfigurieren und auch Reparieren. Außerdem sollen weitere Schnittstellen und Entwicklungsunterstützung angeboten werden. Was weniger dringend benötigt wird sind neue Algorithmen für Routing, die eine Verbesserung von wenigen Prozent versprechen.

2 Semantische Interoperabilität jenseits von Ontologien

von Prof. Werner Kuhn, Universität Münster

2.1 Begriffe

Semantische Interoperabilität ist einfach ausgedrückt die Verbindung von Diensten und Daten. Als Beispiel für solch eine Verbindung wurde die Nutzung von verschiedenen Diensten gezeigt, um bei einem Gasleck in einer Fabrik eine Karte zu erstellen, die die auftretende Gaswolke beschreibt. Dabei werden zuerst die Emissionsrate und der Austrittsort bestimmt, dann wird der naheste Flughafen ausfindig gemacht, um Informationen über den Wind zu erhalten. Aus Emissionsrate und Windinformationen kann schließlich die Gaswolke genau spezifiziert werden und man kann eine Karte erstellen. Probleme treten dabei auf, weil es für die selben Dinge oft unterschiedliche Spezifikationen gibt. Im Beispiel wurde die Windrichtung vom einen Dienst von Ost nach West und vom anderen Dienst von West nach Ost definiert, somit ist das Resultat am Ende falsch. Ein anderes Beispiel ist eine Grünfläche. Diese wird teilweise als Wiese, Acker oder auch Rasen bezeichnet. Es bedarf also einer genauen Spezifikation. Deshalb werden Ontologien verwendet.
Eine **Ontologie** an sich kann man kurz als „Specification of Conceptualization“, also als Spezifikation einer Konzeption, definieren. Im Beispiel mit der Windrichtung ist die Windrichtung an sich das Konzept und die Spezifikation ist die Definition der Windrichtung, also aus welcher Richtung man den Wind misst. Weitere Beispiele sind Legenden von Landkarten aber auch das Periodensystem. Im Gegensatz zu den Ontologien gibt es noch die Folksonomien, dies sind durch die Gemeinschaft geprägte (spezifizierte) Konzepte.
Semantik kann vereinfacht durch das „seimiotische Dreieck“ beschrieben werden. An den Ecken stehen dabei die Begriffe des Konzeptes, des Symbols und der Beziehung. Ein Konzept drückt dabei ein Symbol aus, ein Symbol referenziert eine Beziehung und diese führt wiederum zu einem Konzept.

2.2 Forschung

- „Reference Spaces“: Es werden bestimmte Objekte durch verschiedene Eigenschaften gekennzeichnet. Dabei werden diese Eigenschaften immer weiter aufgeteilt, so dass eine Baumstruktur entsteht. Ein Bespiel sind verschiede Obstsorten, die alle die Eigenschaft des Volumens besitzen. Daraus kann man Skalen entwickeln, die das Obst anhand der Eigenschaft Volumen beschreiben.

- „Similarity“: Hierbei werden Ähnlichkeiten von verschiedensten Objekten ausfindig gemacht. Die Spezifikation ist hierbei immer syntaktisch.

- „Process Grounding“: Begriffe werden in ihre „Atome“ zerlegt um dann Verbindungen zwischen diesen Atomen schematisch zu definieren. Ein Beispiel dafür ist die Verbindung zwischen Boot und Haus. Aus diesen beiden Begriffen kann man das Wort Bootshaus und das Wort Hausboot ableiten. Beide Begriffe haben eine unterschiedliche Bedeutung, obwohl sie aus den gleichen Atomen bestehen. Ein weiteres Beispiel ist die Verwendung von Konzepten. Das Konzept des Transports kann auf die atomaren Konzepte des Weges und der Oberfläche zurückgeführt werden.

3 Stochastic model checking for performance and dependability evaluation

von Prof. Boudewijn R. Haverkort University of Twente

3.1 Leistungs- und Zuverlässigkeitsbewertung

Die Leistungsbewertung klärt Fragen über die Leistungsfähigkeit eines Systems. Man ist zum Beispiel daran interessiert, wieviele Clients ein Server in einer bestimmten Zeit bedienen kann oder wie hoch die Auslastung bei einer bestimmten Anzahl von zugreifenden Clienten ist.
Zuverlässigkeitsbewertung hingegen klärt Fragen über die Verfügbarkeit eines Systems. Beispiele sind Fragen ob das System 99 Prozent des Jahres verfügbar ist, ob es die nächsten 10 Stunden ohne Fehler läuft oder wieviele Jobs ohne Fehler erledigt werden können.
Eine spezielle Form der Bewertung ist die modellbasierte Leistungs- und Zuverlässigkeitsbewertung. Hierbei beruhen die Vorhersagen auf abstrakten mathematischen Modellen. Die Wahl des korrekten Modells ist hierbei enorm wichtig. Man muss entscheiden, was man eigentlich modellieren will und wie genau das Modell sein muss. Zum Erstellen solcher Modelle gibt es spezielle Software, die Ansätze für die Verifikation der Modellen sind dann analytisch, numerisch oder simulativ.

3.2 Stochastische Modelle

Stochastische Modelle werden zum Beispiel durch Markovketten realsiert. Markovketten enthalten Zustände und Zustandsübergänge, welche durch Zeiten oder Wahrscheinlichkeiten charakterisiert werden. Markovketten entstehen dabei oft aus Petrinetzen.
Für Markovketten gibt es einige wichtige Transformationen, die bestimmt Tests erheblich vereinfachen.
Zum einen gibt es die „Uniformisation“. Dabei werden kontinuierlich zeitabhängige Markovketten (CTMC) in diskrete zeitabhängige Markovketten (DTMC) umgewandelt. Danach stehen an den Kanten zwischen den einzelnen Zuständen keine Zeiten mehr, sondern Wahrscheinlichkeiten.
Eine weitere Transformation ist das so genannte „Lumping“. Dabei werden Zustände gleicher Äquivalenzklassen zusammengefügt, so dass sich die Zahl der Zustände meist erheblich verkleinert. Die Äquivalenzklassen hängen dabei vom Modell ab.
Als Beispiel wurde ein System aus 3 Prozessoren und einer „Entscheidungseinheit“ angebracht. Die „Entscheidungseinheit“ legt dabei fest, welcher Prozessor eine Anfrage ausführt. Das System läuft nicht mehr, wenn mehr als 2 Prozessoren ausfallen oder wenn die „Eintscheidungseinheit“ ausfällt. Dieses System kann durch eine Markovkette mit 5 Zuständen realisiert werden. Da es nicht weiter interessiert, welcher Prozessor ausgefallen ist sondern nur wieviele ausgefallen sind, konnten viele Zustände zusammengefasst werden (durch „Lumping“), so dass nur noch 5 Zustände übrig geblieben sind.

3.3 Formale Verifikation

Bei der formalen Verifikation sollen bestimmte Funktionen eines System verifiziert werden. Dies geschieht durch Testen, Simulation oder durch Modelchecking.
Beim Modelchecking wird von einem realen System ein Modell gebildet und zu testende Eigenschaften werden festgelesgt. Beide Spezifikationen werden dann an einen Verifizierer übergeben, welcher dann mit „Ja“ (das System hat die festgelegte Eigenschaft) oder „Nein“ (das System hat

die Eigenschaft nicht) antwortet.
Die Tests erfolgen dabei mit speziellen Logiken (CTL, LTL, ...) und die Ergebnisse werden meist rekursiv berechnet. Durch Erweiterungen der Logiken durch Operatoren für die Zeit (CSL) kann schließlich eine Verbindung zu den Markovketten hergestellt werden. Dabei entsteht oft das Problem, dass nicht lösbare Diffentialgleichungen entstehen. Die Lösung dafür ist die Verkleinerung der Markovketten durch geeignete Transformationen (siehe Abschnitt „Stochastische Modelle“).
Die genannten Logiken beziehen sich weitestgehen auf die Verifikation der Zuverlässigkeit. Es gibt allerdings auch Modelle zur zusätzlichen Verifikation der Leistung (Performance + Dependability: Performability). Dafür gibt es spezielle Modelle (MRM: Markov Reward Model) mit eigener Logik (CSRL).

3.4 Zusammenfassung und Ausblick

Es werden immer neue und ausdrucksstärkere Techniken zur Spezifikation von Leistungs- und Zuverlässigkeitseigenschaften entwickelt. Durch spezielle Tools ist weitestgehend auch eine automatische Verifikation möglich. Allerdings können noch nicht alle Eigenschaften problemlos verifiziert werden, da sich viele Fragestellungen nur numerisch lösen lassen. Deshalb werden die Logiken ständig erweitert und dadurch neue Tools entwickelt.

4 Herausforderungen und Ansätze für Kontextmodelle

von Prof. Bernhard Mitschang, Universität Stuttgart

4.1 Motivation

Dadurch, dass immer mehr Computer von einzelnen Personen im Alltag benutzt werden (ob bewusst oder unbewusst) und durch die vorhandene Technologie lässt sich die physische Welt sehr genau in einer digitalen Welt abbilden. Dazu werden so genannte Kontextinformationen benötigt, welche zum Beispiel über Sensoren, Dokumente oder Datenbanken verfügbar sind. Ein Beispiel solch einer Abbildung ist der Kontostand bei einer Bank. Das eigentlich physische Objekt „Geld“ wird auf einen digitalen Zahlenwert abgebildet. Kontextbezogene Systeme haben weiterhin ein enormes Marktpotential und werden in nahezu allen Anwendungsgebieten genutzt.

4.2 Kontexte

Es gibt 4 verschiedene Kontextarten:

- **geographischer Kontext**: alles, was auf Karten steht
- **dynamischer Kontext**: alles, was sich bewegt
- **Informationskontext**: alles aus der digitalen Welt
- **technischer Kontext**: Zugang, Infrastruktur, Netze

Beispiele für kontextbezogene Anwendungen sind Navigationsdienste, Informationssysteme, Kommunikationsdienste oder Lebensszenarien.
Ein spezieller und in der Praxis interessanter Kontext (zum Beispiel für Navigationsdienste) ist der Kontext der Nachbarschaft. Objekte der physischen Welt werden über Koordinaten beschrieben, man kann damit aber nicht viel anfangen. Deshalb kann man eine Karte darüberlegen und dadurch die Nachbarschaft zu anderen Objekten genauer beschreiben. Somit kann zum Beispiel ermittelt werden, wo die nächste Bushaltestelle ist oder wo das nächste Restaurant ist.

4.3 Umgebungsmodelle

Da sich viele Bereiche der Kontextarten überlappen und von mehreren kontextbezogenen Anwendungen benutzt werden, muss dafür gesorgt werden, dass diese Bereiche ordentlich geteilt werden. Dazu wurde ein so genanntes Umgebungsmodell entwickelt, welches eine Schnittstelle zwischen physischer Welt, digitaler Welt und der kontextbezogenen Anwendung darstellt.
Dabei gibt es lokale Modelle, welche sich gegenseitig überlappen, und globale Modelle, welche die lokalen Modelle nutzen. Bei der Entwicklung der Modelle muss immer ein gesundes Maß zwischen Flexibilität und Interoperabilität gefunden werden.
In den Umgebungsmodellen existieren verschiedenste Objekte mit Attributen. Es sind dabei auch Mehrfachattribute erlaubt. Die Objekte existieren in Objekttypen, welche 2 Schemata definieren. Im Standardschema sind ca. 250 Objektklassen definiert, in denen alle Kontextarten vertreten sind. Das erweiterte Schema enthält Spezialisierungen dieser Standardobjekte.
Die Realisierung dessen erfolgt mittels XML. Hauptgrund ist die Flexibilität (zum Beispiel stellen Mehrfachattribute kein Problem dar) und die Existenz vieler Tools. Es existiert eine eigene Anfragesprache (AWQL) und eine eigene Serialisierungssprache (AWML).

5 Model-based design of provably fault-tolerant systems

von Prof. András Pataricza, Budapest University

5.1 Motivation

Bei den Konzepten des Designs von komplexen Systemen hat sich in letzter Zeit viel geändert. Im Gegensatz zu traditionellen Konzepten (Vorgehen: Spezifikation → Modell → Implementation), wo alle Schritte noch von Menschen ausgeführt werden, bieten neuere Konzepte wie zum Beispiel MDA (Model Driven Architecture, Konzept der OMG, Vorgehen: UseCase → Architektur → Implementation) viele Automatisierungen an, so dass nicht in jedem Schritt mehr Menschen benötigt werden. Weitere Vorteile dieser neueren Konzepte bei den Kosten und bei den Fehlern im Quellcode macht die folgende Tabelle deutlich (Abschätzung nach Prof. Pataricza):

Konzept	**Kostenabschätzung**	**Fehler/kLOC**
Traditionell	100 %	11
UML	42 %	6
MDA	11 %	2,5
MDA und formale Analyse	7 %	0,25

Viele Probleme bei der Automatisierung wie zum Beispiel die Syntaxanalyse, die Codegenerierung und die Ermöglichung von Verfeinerungen sind bereits gelöst, allerdings gibt es noch viele ungelöste Probleme, wie die Modellierung der Semantik, die Zuverlässigkeit oder auch die Performanz.

5.2 Modelltransformationen

Bei der Nutzung von Automatisierungen werden häufig Modelltransformationen genutzt. Ein Tool, welches dieses unterstützt, nennt sich VIATRA (Visual Automated Model Transformation). Dieses Tool ist für die Entwicklungsumgebung Eclipse erhältlich und stellt ein Framework zur Verfügung, welches den kompletten Software-Life-Cycle unterstützt. Man kann dabei in bzw. zwischen verschiedenen Modellsprachen transformieren. Dabei wird das Prinzip der Graph-Transformation genutzt.

5.3 Fehlermodelle

Eine mögliche Modelltransformation ist die Transformation eines Ausgangsmodells in ein Fehlermodell, welches wiederum in ein Zielmodell transformiert wird. Ein Fehlermodell kann dabei formal definiert werden, was aber durch die hohe Komplexität nur eingeschrängt möglich ist, qualitativ definiert werden, indem man den Suchraum reduziert, oder systematisch definiert werden.
Ein Fehler selbst tritt vereinfacht gesagt auf, wenn der Soll-Wert/Zustand eines Programmes nicht mit dem Ist-Wert/Zustand übereinstimmt. In Fehlermodellen wird diese Definition noch weiter verfeinert in kleinere Abweichungen, größere Abweichungen oder sogar Werte/Zustände außerhalb eines definierten Bereiches. Weiterhin wird zwischen statischen und dynamischen Fehlern unterschieden.
Die Auswirkungen von Fehlern lassen sich über Tools simulieren, so dass die beste Lösung für das Vorgehen im Fehlerfall gefunden werden kann. Dabei muss ein gesundes Maß zwischen Effizienz und Zuverlässigkeit gewahrt werden.

6 Smart Earth: Networked Information Management in a Wireless World

von Prof. Karl Aberer, EPF Lausanne

6.1 Wireless Technologie

Prognosen zu Folge werden im Jahr 2017 ca. 1000 „Wireless Devices“ pro Person auf der Erde vorhanden sein. Diese Geräte sind größtenteils unsichtbar und kommunizieren untereinander. Somit erhalten „Wireless Networks“ eine immer größere Bedeutung und man versucht die grundlegenden Prinzipien weiter zu erforschen und dadurch Entwicklungsplattformen, sowohl auf Hardware- als auch auf Softwarebasis, zur Verfügung zu stellen.
Ein klassiches Beispiel eines solchen Netzwerkes ist ein „Wireless Sensor Network“ (WSN). Jeder Knoten des Netzwerkes hat dabei 3 Funktionen: kommunizieren, messen und berechnen. Die einzelenen Knoten sollen dabei möglichst wenig Energie verbrauchen und kostengünstig sein. Ein Beispielanwendungsgebiet für WSNs ist die Umweltüberwachung.
An der EPFL wurde ein Praxistest durchgeführt, bei dem der Wasserkreislauf auf dem Campus durch ein WSN überwacht wurde. Aus den Erfahrungen des Tests konnten einige Gerneralisierungen für WSNs entwickelt werden. Zum einen ein „Sensor Platformkit“, welches Hardware, Software und Tools zum Einsatz von WSNs zur Verfügung stellt, und zum anderen das so genannte „Global Sensor Network“, welches eine Middleware zur Verbindung von WSNs mit dem Internet darstellt.
Abschließend kann man zur Wireless Technologie sagen, dass sie sich im Moment sehr rasant entwickelt. Es existieren zwar noch keine 1000 „Wireless Devices“ pro Person, die Geräte selber sind noch nicht günstig genug und sie sind noch zu groß, aber es ist sehr wahrscheinlich dass sich dies in der nahen Zukunft ändert.

6.2 Smart Earth

Bei „Wireless Networks“ gibt es 3 Hauptprobleme:

1. Sind alle Knoten noch miteinander verbunden?
2. Wie sind die gelieferten Daten zu interpretieren?
3. Welchen Daten kann man trauen?

Smart Earth setzt dabei die physikalische Welt mit 3 Schichten in Verbindung, welche diese Probleme versuchen zu lösen.
Die unterste Schicht stellen „Data Access Networks“ dar. Ein konkretes Beispiel dafür ist ein „Self-Organizing Overlay Network“. In diesen Netzwerken wird die Verteilung der Knoten approximiert und so kann man sehr gut den Nachrichtenaustausch sicherstellen und herausbekommen, welche Knoten nicht mehr erreichbar sind.
Die mittlere Schicht stellen „Semantic Overlay Networks“ dar. Das Problem ist, dass es keine allgemeinen Datenmodelle gibt. Durch „Semantic Gasping“ aber lassen sich verschiedene Modelle, die teilweise den gleichen Zweck erfüllen, in Beziehung setzen. Diese „Mappings“ sind zwar sehr fehleranfällig, aber durch Wahrscheinlichkeiten kann man zeigen, ob ein „Mapping“ korrekt ist.
Die oberste Schicht stellen „Social Networks“ dar. Dabei werden Nachrichten von Knoten bewertet und es wird dadurch die Vertrauenswürdigkeit der Knoten getestet.

7 Vom mobilen zum pervasiven Lernen in einer selbstorganisierenden Umgebung

von Prof. Djamshid Tavangarian, Universität Rostock

7.1 Mobiles Lernen

In den letzten Jahrzehnten hat sich das Lernen bzw. der Lernprozess stark weitereinwickelt: vom entdeckenden Lernen über individuelles Lernen bis hin zum mobilen Lernen.
Das mobile Lernen ist eine Weiterentwicklung des „E-Learning“ und wird von Menschen einer durch Mobilität geprägten Lebens- und Arbeitswelt erwartet. Es funktioniert nach dem Prinzip „anytime, anywhere, anyone“ und unterstützt die großen vier Typen der Mobilität (Nutzermobilität, Dienstemobilität, Gerätemobilität und Sessionmobilität). Beim mobilen Lernen bildet man eine so genannte virtuelle Lerngemeinschaft.
Die Systemarchitektur beim mobilen Lernen sieht folgendermaßen aus: Nach außen hin befindet sich der Client, der auf verschiedene Resourcen und Dienste wie zum Beispiel Repositorys zugreift. Diese sind wiederum nach innen auf verschiedenen Servern gelagert.
Der Erfolg des mobilen Lernens liegt darin, dass man gute Kommunikationsmöglichkeiten hat und dass man einen aktiven Lernprozess betreibt.

7.2 Pervasives Lernen

Der Nachfolder des mobilen Lernens ist das pervasive Lernen. Beim pervasiven Lernen werden allgegenwärtige Dienste in einer Umgebung mit versteckten intelligenten Rechensystemen für den Lernprozess genutzt. Das Ziel ist es dabei, dass diese Dienste genau dort verfügbar sind, wo sie gebraucht werden. Dadurch hat das pervasive Lernen die Eigenschaften der Ubiquität, der Mobilität und der Skalierbarkeit. Die Auswirkung ist, dass die Universität zum Studenten kommt und der Student nicht wie beim mobilen Lernen die Dienste selbst suchen muss.
Die Systemarchitektur besteht beim pervasiven Lernen aus drei Schichten: aus der Nutzerschicht, aus der Kommunikationsschicht (z.B. Intranet) und aus der Diensteschicht (z.B. Speicher, Server, Resourcen und Dienste). Dabei muss neben einer „Service Oriented Architecture“ auch eine Selbstorganisation gewährleistet sein, welche durch die Komplexität des Systems erforderlich ist.
Man kann sich folgende Beispielanwendungen vorstellen:

- Ein Student möchte ein Dokument im Institut ausdrucken. Die pervasive Umgebung findet dann heraus, wo der nächste Drucker steht und druckt das Dokument aus. Der Student bekommt eine Nachricht sobald das Dokument ausgedruckt ist und erhält eine Wegbeschreibung zum Drucker.

- Ein Student möchte sich in ein gewisses Thema einarbeiten. Das System sucht daraufhin Materialien zu dem Thema und liefert diese an den Studenten zurück.

- Eine Synchronisierung von zur Verfügung gestellten Daten (wie zum Beispiel Vorlesungsfolien) mit den Daten auf dem Laptop des Studenten wird vom System automatisch vorgenommen.

Durch pervasives Lernen wird es einige Veränderung für die Studenten aber auch für die Dozenten geben. Die Herausforderung wird die Entwicklung der Lernumgebung sein.

8 CSP - neue Einsichten in ein altes Paradigma

von Prof. Markus Roggenbach, University of Wales Swansea

8.1 CSP

CSP (Communicating Sequential Processes) ist eine Prozessalgebra, die zur Verifikation von Systemen eingesetzt wird. Sie wird unter anderem auch verwendet, um die Fehlertoleranz selbstorganisierender Systeme zu verifizieren. Es wurde zum Beispiel die Fehlertoleranz des Datenmanagementsystems auf der internationalen Raumstation ISS mittels CSP verifiziert.
Es existiert eine festgelegt Syntax für CSP. Man kann Systeme mit einem Alphabet und einer vorgegebenen Menge von Basisprozessen beschreiben (z.B. Start, Stop, Skip). In der Praxis gibt es verschiedene gängige Modelle zur Verifikation von gewissen Eigenschaften. Diese kann man mittels CSP beschreiben. Man kann so zum Beispiel mit dem Modell „Trace“ die Sicherheit oder mit einem Fehlermodell die Fehlertoleranz eines Systems verifizieren.

8.2 CSP-Prover

Zur Unterstützung von Beweisen mittels CSP wurde ein halbautomatischer Theorembeweise, der CSP-Prover, entwickelt. Dieser basiert auf dem Beweisassistenen Isabelle/HOL. Es wird dem Nutzer dabei eine Architektur und eine Oberfläche zur vereinfachten Eingabe der Beweise zur Verfügung gestellt. Weiterhin werden bereits einige Modelle unterstützt wie zum Beispiel das Fehlermodell und teilweise auch das „Trace“-Modell. Es werden vom System verschiedene Fixpunktberechnungsstrategien verwendet.
Mit diesem Tool wurden bereits Beweise für einige Systeme geführt, wie z.B. für ein Kreditkartensystem oder für verschiedene algebraische Gesetze. Allgemein erfolgt der Einsatz in der Industrie, im militärischen Bereich und im wissenschaftlichen Bereich.
Das Ziel zukünftiger Entwicklungen ist es, weitere Modelle in den CSP-Prover zu integrieren und die Analyse paralleler Algorithmen zu ermöglichen.

9 Context-based services - a basis for self-organizing systems

von Prof. Schahram Dustdar, Technische Universität Wien

9.1 Einführung

Durch die Evolution der Technik haben sich in letzter Zeit einige grundlegende Dinge geändert. Endgeräte werden kleiner und günstiger und sind permanent online, die Kommunikation erfolgt oft pervasiv („anytime, anywhere“) und die Netzinfrastrukturen werden offener. Softwarevoraussetzungen sind dadurch dezentraler und nicht mehr vorab erfassbar, Prozesse (z.B. Geschäftsprozesse) benötigen Daten von mehreren verschiedenen Systemen und auch die Interaktionen zwischen den Menschen selbst haben sich von lange zusammenarbeitenden stabilen Teams hin zu flexiblen kurzlebigen Teams gewandelt. Dies liegt an der immer größer werdenden Mobilität, der Kurzlebigkeit und der vermehrten Interaktion des Menschen mit seiner Umgebung. Um diesen Entwicklungen nachzukommen ist die Entwicklung von neuen Konzepten enorm wichtig.

9.2 Konzepte für kontextbasierte Dienste

Ein Ansatz ist das „Service Oriented Computing“. In solchen „Service Oriented Systems“ existieren aktive Objekte, die einen Dienst anbieten und nach Möglichkeit auf unvorhergesehene Ereignisse selbständig reagieren. Dadurch ist allerdings die Hardware- und Softwarearchitektur nicht mehr im Voraus exakt planbar und die Infrastruktur des Systems wird sehr komplex, da die Abhängigkeiten von Teilsystemen nicht mehr fixiert sind. Den Begriff der Service-Orientierung kann man auch auf die Wirtschaftswelt anwenden (Stichwort Dienstleistungsgesellschaft).
Ein Service selbst wird als wohldefinierte, einfach zu bedienende standardisierte Schnittstelle definiert. Er ist eigenständig, sollte ständig verfügbar sein, ist unabhängig vom Kontext des Benutzers (für einen Kreditkartenservice ist es uninteressant, was eine Person kauft sondern nur dass sie etwas kauft und dadurch ein gewisser Betrag abgebucht werden muss) und hat gewisse „Quality of Service“-Attribute (z.B. Performanz, Verfügbarkeit, Kosten). Neue Services können aus der Komposition vorhandener Services entstehen.
Der Kontext spielt bei solchen Systemen eine wichtige Rolle. Eine Zukunftsvision ist zum Beispiel ein System, welches den richtigen Kredit für eine bestimmt Person auswählt. Dabei sind gewisse Kontextinformationen wie zum Beispiel der Verdienst der Person oder ähnliches von großer Bedeutung.
Es gibt verschiedene Ansätze um den Kontext in solche Systeme bzw. Dienste zu integrieren bzw. diese so anzupassen, dass Kontextinformationen genutzt werden können. Um diese Informationen sammeln zu können, wird die Welt in so genannt Sichten eingeteilt wie z.B. in eine örtliche Sicht, eine organisierte Sicht oder eine Interaktionssicht. Anschließend werden diese Sichten in Beziehung gesetzt und die daraus gewonnenen Informationen sollen beim Entwurf neuer kontextbasierter Dienste helfen. Diese Informationen werden dann für das Konzept der „Autonomic Service Adaption“ genutzt. Des Weiteren können für dieses Konzept Aktivitätsmuster verwendet werden, die Aktivitäten Dienste zuordnen, oder es kann eine Analyse der Benutzung von Services erfolgen um herauszufinden, welche Services häufig hintereinander ausgeführt werden. Es existieren außerdem bereits Kontextmodelle, die das Zusammenspiel der Kontexte Ort, Gerät, Benutzer und Aktivität modellieren. Diese Modelle sind aber noch unvollständig und müssen erweitert werden.

10 Time Synchronization in Wireless Ad-Hoc Networks

von Prof. Reinhard Gotzhein, Technische Universität Kaiserslautern

10.1 Grundlagen der Zeitsynchronisation

Zuerst musste geklärt werden, was der Begriff der Zeit überhaupt bedeutet. Die Bedeutung haben sowohl Naturwissenschaftler als auch Philosophen hinlänglich diskutiert. Weiterhin sollte es möglich sein, Zeit in einheitlichen Größen zu messen. Dies geschieht mit Uhren verschiedenster Art (z.B. Sonnenuhr, Sanduhr, Atomuhr). Als Einheiten wurden bestimmte Konstrukte wie Sekunden, Minuten, Stunden, usw. eingeführt. Für eine Sekunde gibt es dabei eine ganz genaue wissenschaftliche Definition, auf welche sich dann weitere Zeiteinheiten beziehen. Um eine gemeinsame Basis für die Zeitmessung zu haben, wurden zudem verschiedene Referenzzeiten eingeführt (z.B. UT, UTC, TAI).
Das Ziel der Zeitsynchronisation ist es nun, keine bzw. eine hinreichend kleine Abweichung zwischen der Zeit eines Objektes und einer Referenzzeit zu haben. Im Anwendungsgebiet der Sensornetzwerke ist dies zum Beispiel erforderlich, um die Reihenfolge der von verschiedenen Knoten registrierten Ereignisse festzustellen. Dabei werden so genannte Zeitstempel benutzt.
Die Anwendungsgebiete für Zeitsynchronisation sind weitläufig. Man benötigt sie in Echtzeitsystemen, in denen die Ergebnisse von der Zeit abhängen, oder in Sensornetzwerken, in denen Daten zusammengetragen werden, welche nach zeitlichen Aspekten ausgewertet werden müssen. Ein weiteres Anwendungsbeispiel sind vernetzte Kontrollsysteme wie beispielsweise ein System, in dem sich ein Stab auf einem bewegten Wagen befinden und nicht umfallen soll. Die angebrachten Sensoren müssen dabei zeitsynchronisert sein um den Bewegungen des Stabes entsprechend gegenwirken zu können.
Im folgenden werden einige wichtige Konzepte bei der Zeitsynchronisation beschrieben. Da die Übersetzungen ins deutsche oft ungenau sind, werden die englischen Bezeichnungen verwendet:

- **real time:** stellt eine Referenzzeit für ein System dar
- **local time:** Zeit irgendeines Objektes im System, im Beispiel des Ad-Hoc Netzwerkes eines Knotens
- **clock-offset:** ist die Differenz zwischen zwei Zeiten (entweder zwei lokalen Zeiten oder zwischen lokaler und realer Zeit)
- **clock-frequency:** ist die erste Ableitung der Zeit (für die reale Zeit ist diese gleich 1)
- **clock-skew:** ist die Different zwischen zwei „clock-frequencies“
- **clock-drift:** ist die zweite Ableitung der Zeit (für die realte Zeit ist diese gleich 0)

Da sich die folgenden Betrachtungen zumeist auf „Wireless Ad-Hoc Networks“ beziehen folgt nun eine kurze Definition. Ein „Wireless Ad-Hoc Network“ ist ein selbstkonfigurierendes Netzwerk, welches aus statischen oder mobilen Knoten besteht welche untereinander kommunizieren. Eigenschaften sind eine dynamische Topologie oder eine variierende Kanalqualität. Es existieren gewisse Schwierigkeiten in diesen Netwerken, welche z.B. durch Kollisionen oder durch assymetrische Verbindungen entstehen.
An die Zeitsynchronisation in solchen Netzwerken werden gewisse Anforderungen gestellt. Es soll

eine hohe Genauigkeit erreicht werden, entweder bezüglich einer Referenzzeit außerhalb des Systems (externe Zeitsynchronisation) oder bezüglich der lokalen Zeit aller anderen Knoten im Netzwerk (interne Zeitsynchronisation). Dies soll mit einer möglichst geringen Komplexität erfolgen, das heißt es soll wenig Energie und wenig Rechenleistung verbraucht werden. Weiterhin soll die zum Synchronisieren benötigte Zeit möglichst klein sein. Bei „Wireless Ad-Hoc Networks“ müssen zudem die dort auftretenden Schwierigkeiten berücksichtigt werden.

10.2 Zeitsynchronisationsalgorithmen

- **Remote Clock Reading Method:** Bei dieser recht einfachen Methode sendet ein Client eine Nachricht zum Zeitpunkt t_0 an einen Server, dieser versieht die Nachricht mit einem Zeitstempel t_{TS} und schickt sie anschließend an den Client zurück. Dieser erhält die Nachricht inklusive Zeitstempel dann zum Zeitpunkt t_0. Durch einfache Berechnungen $(t_{TS} - \frac{(t_1 - t_0)}{2})$ lässt sich so die Zeit des Servers auf dem Client ermitteln. Diese Methode ist zum einen sehr ungenau und zum anderen entsteht ein hohes Kommunikationsaufkommen.

- **Network Time Protocol (NTP):** Beim NTP handelt es sich um den Standard für Zeitsynchronisation über das Internet. Dieses Protokoll ist allerdings für „Wireless Ad-Hoc Networks“ nicht geeignet, da es zum einen zu komplex ist und zum anderen die Knoten immer online sein müssten, wodurch ein zu hoher Energiebedarf entsteht.

- **Tiny-Sync Protocol (TSP):** Dieses Protokoll weist gewisse Ähnlichkeiten zur „Remote Clock Reading Method“ auf. Auch hier sendet ein Sender A eine Nachricht zum Zeitpunkt t_0 an Empfänger B. Sobald dieser die Nachricht erhält (Zeitpunkt t_1) schickt er eine Nachricht zurück an Sender A, welcher diese zum Zeitpunkt t_3 erhält. An jedem der Zeitpunkte wird ein Zeitstempel angelegt. Die 3 Zeitstempel werden Datenpunkt genannt. Aus einem Datenpunkt lässt sich der „clock-offset“ berechnen und aus mehreren der „clock-skew“. Die Vorteile dieses Protokolls sind der geringe Speicherbedarf und der geringe Rechenaufwand (nur die 4 „besten“ Datenpunkte werden für gute Abschätzungen benötigt). Nachteile sind eine notwendige Hierarchie unter den einzelnen Knoten im Netzwerk, eine nur paarweise zwischen den Knoten stattfindende Synchronisation, eine geringe Robustheit gegenüber Topologieänderung und ein hohes Kommunikationsaufkommen.

- **Reference Broadcast Synchronization (RBS):** Bei diesem Protokoll sendet ein Knoten an alle Knoten in seiner Reichweite eine Referenznachricht („Broadcast Message“). Die Empfänger merken sich die Empfangszeit und tauschen sich untereinander aus. Somit lässt sich der „clock-offset“ bestimmen. Um den „clock-skew“ zu bestimmen muss man diesen Vorgang wiederholen. Durch dieses Vorgehen entstehen im Netzwerk mehrere Partitionen, für welche verschiedene Zeitzonen definiert werden. Dabei können einzelne Knoten ihre Zeit zwischen den Zeitzonen konvertieren. Vorteil ist eine sehr hohe Genauigkeit. Nachteile sind ein hohes Kommunikationsaufkommen und eine geringe Robustheit gegenüber Topologieänderung.

- **Timing-Sync Protocol for Sensor Networks (TPSN):** Dieses Protokoll funktioniert in zwei Phasen. In der ersten Phase wird eine Hierarchie im Netzwerk aufgebaut, so dass jeder Knoten seinen eigenen Standpunkt, also sein Level und seinen Referenzknoten, genau kennt. In der zweiten Phase, der Synchronisationsphase, wird eine paarweise Synchronisation für Knoten auf den Leveln `i` und `i+1` durchgeführt. Vorteile sind eine hohe Genauigkeit und eine hohe Robustheit gegenüber Kollisionen. Nachteile sind erneut ein hohes Kommunikationsaufkommen und eine geringe Robustheit gegenüber Topologieänderung.

Bewertet nach der Genauigkeit sind die Protokolle RBS und TSPN sehr genau, wobei RBS am genausten ist. Das Protokoll TSP weist nur eine mittelmäßige Genauigkeit auf. Bezüglich der Komplexität sind die drei genannten Protokolle größtenteils identisch.

10.3 Zusammenfassung

Zusammenfassend kann man sagen, dass bereits einige Algorithmen für Zeitsynchronisation existieren. Diese haben zum Teil eine hohe Genauigkeit und eine ordentliche Komplexität. Allerdings gibt es keine deterministischen Grenzen für die Konvergenz und die Robustheit gegen Topologieänderungen ist auch nicht gegeben. Es existiert also momentan noch kein perfekter Zeitsynchronisationsalgorithmus für „Wireless Ad-Hoc Networks“.

11 Service Oriented Architectures: A Technical Overview

von Prof. Mike P. Papazoglou, Tilburg University

11.1 Service Oriented Architecture

Die „Service Oriented Architecture“ (SOA) ist eine Architektur, die einzelne ungebundene Services wie zum Beispiel Web-Services nutzt, um die Anforderungen von Geschäftsprozessen zu unterstützen.
Ein Web-Service stellt Funktionalität bereit, die unter anderem in der Geschäftswelt benötigt wird. Ein einfaches Beispiel ist die Abfrage von Aktienkursen. Einzelne Web-Services können zu komplexeren Systemen zusammengefasst werden, um Geschäftsprozesse wie zum Beispiel die Logistikabläufe eines Unternehmens zu beschreiben. Web-Services können von Anwendungen benutzt werden und basieren auf offenen Standards.
Resourcen eines Netzwerkes in einer SOA-Umgebung werden als unabhängige Services zur Verfügung gestellt, auf die ohne Wissen über die zugrundeliegene Implementation zugegriffen werden kann. Eine SOA-Umgebung funktioniert folgendermaßen: Es existiert ein Service-Broker, ein Client und ein Service-Provider. Der Service-Provider bietet Services über einen Service-Broker an, an den der Client Suchanfragen stellt. Anschließend greift der Client nach erfolgreicher Suche auf den Service des Service-Providers zu. Der Service-Provider stellt die Implementation und die Businesslogik zur Verfügung, was einer logischen und einer physischen Schicht entspricht. Der Client kann schließlich auf den Service über ein Interface zugreifen.

11.2 Enterprise Service Bus

Der „Enterprise Service Bus“ (ESB) ist eine standardbasierte Schnittstelle, die nachrichtenorientierte Middleware-Funktionalität zur Verfügung stellt um heterogene Systeme bzw. Komponenten zu verbinden. Damit lässt sich SOA entwickeln und einsetzen. Es werden dabei Charakteristiken verschiedener Middleware-Konzepte (ORB, RPC, MOM) in einem Paket vereinigt. Weiterhin werden verschiedene Standards wie zum Beispiel WSDL und SOAP und verschiedene Plattformen unterstützt. Die Kommunikation zu den Clients erfolgt dabei über WSDL und SOAP und die Kommunikation zur Middleware über spezielle Adapter. Man bezeichnet den ESB auch als eine neue Generation der Middleware.
Ein ESB stellt unter anderem folgende Funktionalität bereit:

- Kommunikation zwischen Services
- dynamische Konnektivität
- Sicherheitskonzepte
- Skalierbarkeit
- Monitoring

Zusammenfassend kann man sagen, das der ESB ein sehr mächtiges Konzept zur Erzeugung von SOA ist. Es werden verschiedenste Standards unterstützt und es ist eine Interaktion von verschiedenen Plattformen möglich. Der ESB kann somit für viele verschiedene Szenarien verwendet werden.

12 Adaptionsmechanismen zur qualitätsgarantierenden Datenstromanalyse

von Prof. Wolfgang Lehner, Technische Universität Dresden

12.1 Einführung

Die Analyse von Datenstömen findet in vielen Bereichen Anwendung, zum Beispiel bei der Verkehrsanalyse oder bei Wetteranalysen. Dabei ist es oft notwendig, Umgebungsdaten von Sensornetzwerken zu erfassen und zu analysieren. Somit kann man beispielsweise bei Produktionsabläufen gewissen Ereignissen entgegenwirken. Unter Umständen benötigt man dabei Qualitätsgarantien. Man unterscheidet dabei zwischen zeitbasierter und inhaltsbasierter Qualität. Bei der zeitbasierten Qualität möchte man Garantien für die Verzögerung, das ist die Zeit zwischen dem Messen und der Reaktion, bzw. für die Datenrate geben.

12.2 Datenstromverarbeitung

Die Datenstromverarbeitung erfolgt folgendermaßen: Man stellt unter der Nutzung von Datenströmen Anfragen, welche Qualitätsanforderungen erfüllen sollen, berechnet daraufhin die nötigen Resourcen und stellt anschließend einen Abarbeitungsplan auf. Für die Anfragen werden Operatoren verwendet. Bei der Verkettung von einzelnen Operatoren soll gewährleistet sein, dass permanent Daten zur Verarbeitung vorhanden sind. Der Produzent benötigt dabei eine Vorlaufzeit, um für den Konsument bereits Daten im Voraus zu berechnen. Diese Daten werden dann in Puffern zwischengespeichert. Die Vorlaufzeit und die Puffergröße müssen dabei so gewählt werden, dass das gegeben Qualitätsziel eingehalten wird. Man benötigt für die Berechnung dieser Werte Datenraten und Selektivitäten, welche für die einzelnen Operatoren verschieden sind. Schließlich kann man vom Qualitätsziel aus zurückrechnen und die Werte bestimmen bzw. anpassen.
Optimierungsmöglichkeiten bieten generalisierte Operatoren (das sind Operatoren die sich aus mehreren anderen Operatoren zusammensetzen) oder effiziente Speicherstrukturen für Puffer.
Aktuelle Forschungsschwerpunkte liegen in der Entwicklung von verteilten Datenstrommanagementsystemen, in der Realisierung von Fehlertoleranz und in der Unterstützung von Multicore-Systemen.