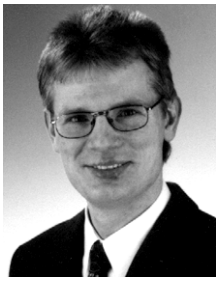
K.G. Saur Verlag, München, 2005

J. Fromm und M. Zapf

„Selbst“-Eigenschaften in verteilten Systemen

Jochen Fromm ist wissenschaftlicher Mitarbeiter an der Universität Kassel im Fachgebiet Verteilte Systeme, wo er sich mit Selbstorganisation und Selbstmanagement in verteilten Systemen beschäftigt. Sein Diplom in Physik erhielt er von der Universität Münster. Anschließend arbeitete er als Anwendungsentwickler im Bereich Software-Entwicklung.



Dr. *Michael Zapf* ist wissenschaftlicher Assistent an der Universität Kassel im Fachgebiet Verteilte Systeme. Seine Forschungsschwerpunkte liegen auf autonomen vernetzten Systemen und verteilt-kooperativer Informationsverarbeitung. Während seiner Promotionsphase an der Goethe-Universität Frankfurt/Main war er im Forschungsfeld der mobilen und autonomen Softwareagenten tätig und spezialisierte sich auf die Kommunikation und Vermittlung von Agenten auf Basis hierarchischer Typsysteme. Nach seiner Promotion war er drei Jahre lang am Fraunhofer-Institut für Sichere Informationstechnologie beschäftigt und arbeitete dort an sicheren agentenorientierten Kollaborationsplattformen.

ZUSAMMENFASSUNG

Unter dem Begriff „Selbst“-Eigenschaften werden Charakteristiken von Systemen zusammengefasst, denen selbstorganisierende Eigenschaften zugesprochen werden. Diese Eigenschaften können in mannigfaltiger Weise klassifiziert werden.

Selbst-Organisation ist in großen, verteilten Systemen von entscheidender Bedeutung. Für diese Systeme stellen eine zentrale Verwaltung und Organisation in vielen Fällen keine optimale Lösung dar. So ist zu erwarten, dass „Selbst“-Eigenschaften in diesen Umgebungen zunehmend in Erscheinung treten.

In diesem Beitrag sprechen wir neben den Gründen und Konsequenzen die Probleme und Perspektiven an, beleuchten den aktuellen Stand von der Terminologie und den Prinzipien bis hin zu modernen Ansätzen und Methoden, und weisen anschliessend kurz auf die zukünftigen Perspektiven hin.

1 EINLEITUNG

Systeme mit sehr hohem Bedarf für Autonomie, Selbst-Organisation, Selbst-Verwaltung und andere „Selbst“-Eigenschaften bestehen aus einer Vielzahl von Komponenten, die entweder zu klein und unzugänglich sind (Ubiquitous Computing), zu komplex und zu zahlreich (Internet-Applikationen mit tausenden von Serven), oder zu weit entfernt (etwa Raum-Sonden im Weltall, siehe das Projekt „Autonomous Nano Technology Swarm“ der NASA [1]), sodass die manuelle Verwaltung zu aufwändig, schwierig und langsam ist.

Bei leistungsfähigen, neuartigen Internet-Applikationen auf globaler Ebene, ist die manuelle Verwaltung und Administration von tausenden von Servern nicht mehr möglich. Proprietäre Systeme von Google und Amazon benutzen massive Replikation – Cluster tausender PCs – und besitzen bereits „Selbst“-Eigenschaften, etwa Selbst-Beobachtung (Self-monitoring, „Continuous inspection“ und Selbst-Heilung (Self-healing, „Continuous refresh“) [2].

Bei Ubiquitous/Pervasive Computing („UbiComp“) [3] und zukünftigen Sensornetzwerken sieht man die mikroskopische IT kaum noch und kann sie daher auch wegen der großen Anzahl an verteilten Einheiten nicht mehr explizit managen. In der Robotertechnik ist Autonomie ein bedeutendes Thema, gerade wenn die manuelle Steuerung und Verwaltung aufgrund von großen Entfernungen und Zeitverzögerungen nicht möglich ist.

Mobile Ad-hoc-Netze (MANETs) [4] sind per Definition selbstkonfigurierend. Weil mobile Geräte ständig und überall neu hinzukommen und verschwinden können und sofort eine Verbindung zueinander aufbauen können, ist außerdem eine manuelle, starre Verwaltung nicht möglich. Die einzelnen Knoten müssen das Netz selbst erkunden und das Routing wegen ständigen Topologiewechsels selbst übernehmen. Die Sicherheit in diesen Netzen ist problematischer als in festen Netzwerken, da keine klare äußere Grenze existiert.

Gute Gründe für Selbst-Organisation, Selbst-Verwaltung und Selbst-Management lassen sich umso eher finden, je stärker die Ressourcen eines Systems verteilt sind und je eigenständiger diese Komponenten agieren müssen.

Die grundlegenden Aufgaben der Verwaltung und der Organisation wird für diese Systeme zunehmend schwierig und problematisch. Wie werden solche Systeme und Netze in Zukunft zu beherrschen sein? Wie kann man die Vorgänge und Prozesse kontrolliert steuern?

Auf der Suche nach Ansätzen stößt man in der Biologie in großer Zahl auf komplexe Systeme und Netzwerke solcher Art. Jeder Organismus ist ein komplexes Gebilde, bestehend aus strukturellen Bestandteilen, den Organen, die ihrerseits aus Zellen bestehen. Zellen wiederum verfügen selbst über ihren eigenen Mikrokosmos. Und trotz all dieser komplexen Zu-

sammenhänge laufen die Prozesse der Natur offensichtlich nahezu fehlerfrei und selbständig ohne Organisator oder Programmierer ab – anders als die uns bekannten IT-Systeme, die uns mit einer nicht enden wollenden Folge von Fehlern und deren Beseitigung beschäftigen.

Unser Dilemma ist, dass wir zwar von wundersam funktionierenden biologischen Systemen umgeben sind, dass wir aber bislang nicht verstehen, wie wir uns deren Mechanismen für die IT zunutze machen können. Es ist klar, dass wir eine Form von Selbst-Organisation und Selbst-Verwaltung in großen verteilten Systemen benötigen, doch wie und in welcher Weise das am besten geschehen soll, bleibt vorerst im Dunkeln.

In diesem Artikel erläutern wir die Terminologie, stellen die grundlegenden Prinzipien und Phänomene der Selbst-Organisation vor, beleuchten die Probleme und Perspektiven, und erörtern die Möglichkeiten von autonomen Systemen mit „Selbst“-Eigenschaften. Dabei gehen wir schrittweise von der Theorie zur Praxis über: nach der Terminologie und Theorie folgen die üblichen Ansätze und Methoden, und schließlich einige konkrete Projekte und praktische Anwendungen.

Wir stellen zunächst die Begriffswelt der Selbstorganisation und der Selbst-Eigenschaften vor (Kapitel 2). Hier wird deutlich, dass es trotz vieler Prinzipien und Begriffe keine einheitliche Theorie gibt. Die Terminologie und Grundprinzipien werden noch sehr kontrovers diskutiert. Selbst konkrete Verfahren und Projekte, die sich mit der Realisierung dieser Eigenschaften befassen (Kapitel 3 und 4), lassen allgemein akzeptierte Vorgehensweisen vermissen.

Die Vielfalt der Begriffe mag verwirrend erscheinen, indes spiegelt sie die Situation der Forschung auf diesem recht jungen Gebiet wieder. Im Kopf sollte man stets die Frage behalten, ob und wie Selbstorganisation mit den vorgestellten Methoden effektiv erreichbar wird.

2 TERMINOLOGIE UND PRINZIPIEN

2.1 Selbst-Organisation

Als Informatiker hofft man durch Betrachtung und Imitation von natürlichen Systemen zu neuartigen biologisch inspirierten Systemen und Verfahren zu gelangen, so wie bei den beiden klassischen Beispielen der neuronalen Netze und evolutionären Algorithmen. Wie der Ingenieur in der Bionik [6] möchte man die besten Prinzipien verwenden, die aus der Biologie abgeleitet werden können.

Dabei gilt es zunächst die Hürde der unterschiedlichen Bedeutungen zu überwinden. Die Begriffe *Selbst-Organisation* und *selbstorganisierendes System* sind nicht klar und eindeutig definiert. Selbst-Organisation ist ein sehr allgemeines, mehrdeutiges und interdisziplinäres Wort, dessen genaue Bedeutung vom jeweiligen Kontext abhängt.

Für den **Biologen** sind selbst-organisierende Systeme durch einen dezentralen Musterbildungsprozess charakterisiert, der auf Schwarm-Intelligenz, Schwarm-Verhalten, Stigmergie und Emergenz zurückzuführen ist – mit anderen Worten: Systeme, in denen es *Organisation ohne Organisator* gibt. Die Biologie ist vor allem an Prinzipien und Prozessen interessiert, um die Systeme in der Natur besser verstehen und klassifizieren zu können. In der Regel bestehen selbstorganisierende biologische Systeme aus Schwärmen, Rudeln oder anderen sozialen Gruppen, vor allem zu finden unter sozialen Insekten wie Ameisen, Termiten und Wespen [7].

Für den **Chemiker** sind selbstorganisierende Systeme vor allem Systeme mit der Fähigkeit zum Selbst-Aufbau (Self-Assembly)[8] oder Autokatalyse. Wie der Biologe und Physiker ist er fasziniert von Muster- und Strukturbildung in so genannten Reaktions-Diffusions-Systemen [7].

Für den **Physiker** sind selbstorganisierende System in der Regel Systeme im kritischen Zustand oder beim Phasenübergang, er spricht dann häufig von selbst-organisierender Kritikalität [9]. Der Physiker konzentriert sich auf einfache Modelle wie etwa Sandhaufen und Netzwerke, weil es dort messbare und fassbare Größen gibt, die man durch mathematische Gleichungen exakt beschreiben kann. Aus thermodynamischer Sicht betrachtet er selbstorganisierende Systeme als offene Systeme, die Ordnung importieren und Entropie (Unordnung) exportieren.

Der **Philosoph**, **Psychologe** und **Soziologe** benutzt den Begriff häufig in einem allgemeinen und abstrakten Sinn und wendet ihn dabei auf ganze Gesellschaften, das Gehirn oder gar das Universum als Ganzes an [10]. Je größer und komplexer das betreffende Objekt ist, desto weniger kann er allerdings etwas Genaues über die tatsächlichen Vorgänge und Prozesse sagen.

Für den **Informatiker** sind selbstorganisierende Systeme solche Systeme, die ohne zentrale Steuerung auskommen und selbstständig auf Veränderungen und Ereignisse in ihrer Umgebung reagieren. Im Unterschied zu starren Automaten und automatischen Systemen reagieren sie flexibel und anpassungsfähig; nur die Selbst-Verwaltung erfolgt automatisch. Solche Systeme sind zum großen Teil noch Vision [20]. Sie erscheinen attraktiv, weil sie Administratoren und Software-Entwicklern uninteressante und repetitive Arbeit ersparen und abnehmen können, indem sie sich selbst organisieren und verwalten. Meist bleiben diese Visionen einer detaillierteren Betrachtung schuldig.

Zurzeit versucht man sich dem Thema durch die Realisierung diverser einzelner „Selbst“-Eigenschaften zu nähern: selbstheilend, selbst-konfigurierend, selbst-optimierend und Ähnliches [20, 29]. Weil man zunehmend auf menschliches Eingreifen verzichten möchte oder muss, strebt man – neben selbständiger Verwaltung – für IT-Systeme die folgenden Eigenschaften an [11]:

- Robustheit und geringe Wartungskosten
- Skalierbarkeit und hohe Performanz
- Anpassungsfähigkeit
- Autonomie

Robustheit ist eine der Hauptunterschiede zwischen traditioneller Software und selbstorganisierenden Systemen. Ein Programm funktioniert sofort nicht mehr, wenn beliebige Codezeilen entfernt oder einzelne Variablen geändert werden, aber ein selbstorganisierendes System (wie ein Sensornetz, neuronales Netz, MANET oder Multi-Agenten-System) funktioniert in der Regel auch dann, wenn beliebige Elemente entfernt werden oder ausfallen.

Skalierbarkeit und Anpassungsfähigkeit sind weitere wesentliche Merkmale, die man von selbstorganisierenden System erwartet. Man könnte meinen, beide Eigenschaften ergeben sich in dezentralen Systemen mit vielen autonomen Einheiten wie in der

Natur von selbst. Doch wie wir später deutlich machen, ist der Preis hierfür, dass man in der Regel auf etwas verzichten muss, etwa auf im IT-Umfeld bisher selbstverständliche Dinge wie Zuverlässigkeit, Vorhersagbarkeit, Verständlichkeit und Kontrolle.

Die vielen verschiedenen Definitionen und Vorstellungen nähren die Vermutung, dass das Grundprinzip – sofern eines existiert – noch nicht richtig verstanden ist. Für den praktischen Anwender gibt es ein undurchschaubares Dickicht von „Selbst"-Eigenschaften [27, 28, 29, 30]. Für den Theoretiker stellt sich die Frage, wie man zu einer grundlegenden Lösung des Problems „Selbst-Organisation in verteilten Systemen" gelangen kann. Beide stehen vor dem Problem, dass der Begriff Selbst-Organisation entweder vage definiert oder sehr mehrdeutig ist. Es gibt weder eine einheitliche Theorie komplexer selbstorganisierender Systeme, noch ein Standardverfahren, um solche Systeme konstruieren zu können.

2.2 Prinzipien und Phänomene

Ein Verständnis der bereits gut bekannten Prinzipien und Phänomene ist hilfreich, um zu einer Aufklärung des Problems zu gelangen. Hinter all den eben genannten unterschiedlichen Definitionen und Sichtweisen verbergen sich einige zentrale Prinzipien und Phänomene, die in natürlichen Systemen wirksam sind.

- **Feedback**: Rückkopplungsschleifen oder Regelkreise zur Steuerung des Verhaltens
- **Emergenz**: Auftauchen von Strukturen und Mustern auf makroskopischer Ebene durch mikroskopische Interaktionen
- **Schwarmbildung**: Dezentrale Akkumulation und Aggregation durch direkte Interaktion
- **Stigmergie und Schwarmintelligenz:** Dezentrale Stimulation und Steuerung des Verhaltens durch indirekte Interaktion mit der Umgebung
- **Skalenfreie Netze**: komplexe Netze, in dem es wenige Knoten mit hohem und viele Knoten mit geringem Verknüpfungsgrad gibt.

Dies sind die zentralen Konzepte, obwohl natürlich nicht auszuschließen ist, das möglicherweise noch weitere, bislang völlig unbekannte Prinzipien und Gesetze existieren. Abb. 1 stellt die drei komplexeren Prinzipien der Emergenz, Schwarmbildung und Schwarmintelligenz dar.

Bei Systemen mit Selbstverwaltung findet man im System explizit eine Trennung in aktive Elemente (Organisator, Verwalter, Manager, Agenten) und passive Elemente (zu organisierende und verwaltende Ressourcen, Komponenten, Dienste). Selbstverwaltung in autonomen Systemen ist der Versuch, die Selbstorganisation von natürlichen Systemen zu imitieren. Die grundlegenden Bausteine eines autonomen Systems sind autonome Elemente, bestehend aus einem verwalteten Element und einem autonomen Manager/Agenten. Wegen dieser Trennung beeinflussen und stören sich die Rückkopplungsschleifen nicht direkt gegenseitig. Allerdings können sich dafür zwischen den dazugehörigen Managern, Verwaltern oder Agenten Konflikte ergeben, beispielsweise das Problem der Oszillation oder chaotischen Verhaltens, das sich automatisch aus mehreren miteinander verkoppelten und verwobenen Regelkreisen ergibt [21].

Im Unterschied zur Selbstverwaltung im IT-Umfeld gibt es bei echter Selbstorganisation *im biologischen Sinn* keinen Organisator oder Manager [5, 7]. Dafür zeichnen sich solche selbstorganisierenden Systeme in der Regel durch *emergente Eigenschaften* aus.

Das Phänomen der *Emergenz* [12, 13] beschreibt das Auftauchen von Strukturen und Mustern auf makroskopischer Ebene durch mikroskopische Interaktionen. In komplexen Systemen mit emergenten Eigenschaften gibt es einen deutlichen Unterschied zwischen individuellen und gesamten Eigenschaften, sowie zwischen lokalem und globalem Verhalten.

Stigmergie [7] bezeichnet den Einsatz von biochemischen Botenstoffen und indirekter Kommunikation durch die Umgebung. Die autonomen Einheiten oder Agenten werden durch das Produkt ihrer Arbeit (wie ein Nest oder die Pheromonspur) zu weiterer Arbeit stimuliert und angeleitet.

Zwei weitere Konzepte sind für selbstorganisierende Netzwerke besonders wichtig, die so genannten *Kleine-Welt-Netzwerke* („Small world networks") [14] und *skalenfreie Netze* („Scale-free networks") [15]. Beides sind Netzwerke, bei denen einige wenige Knoten (Hubs oder Superknoten) sehr viele Verbindungen aufweisen, während ein Großteil der übrigen Knoten relativ wenige Beziehungen zu anderen Knoten hat. Beide Arten von Netzwerken sind recht häufig anzutreffen, entstehen quasi wie „von selbst", sind gut skalierbar, und zeichnen sich durch einen geringen Durchmesser (der kürzeste Weg zwischen den am weitesten entfernten Knoten) sowie kleine durchschnittliche Pfadlänge aus. Außerdem haben sie eine relativ hohe Robustheit und Ausfallsicherheit, sofern man zufällige Knoten entfernt.

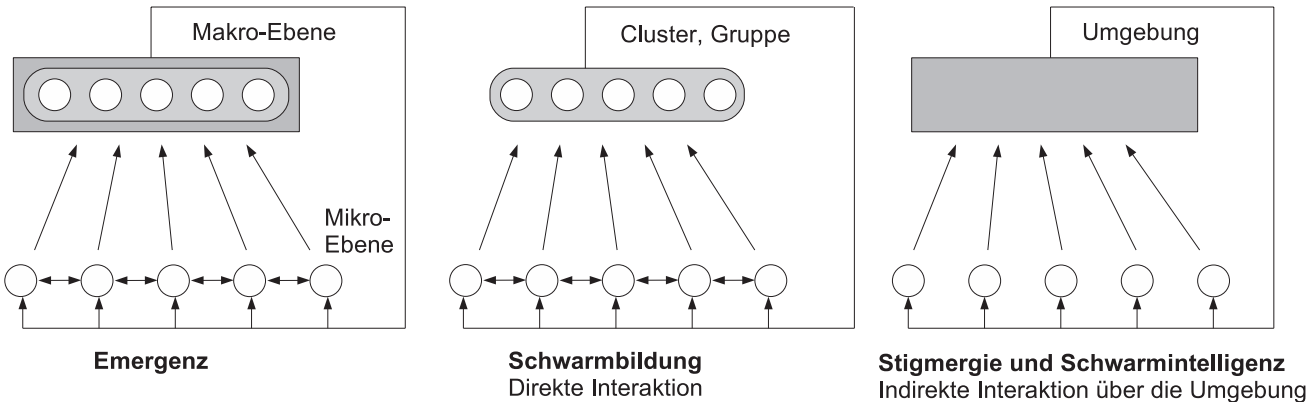


Abb. 1 Prinzipien der Selbstorganisation.

Zwischen all diesen Begriffen und Phänomenen gibt es natürlich Überschneidungen und Überlappungen; skalenfreie Netze zeigen häufig auch eine Kleine-Welt-Eigenschaft und umgekehrt. Emergenz ist unter anderem die Grundlage von Schwarmintelligenz und Schwarmbildung. Ob diese Prinzipien als grundlegende Bausteine von selbstorganisierenden Systemen und Netzen dienen können, ist noch nicht vollständig geklärt. Einige Überlegungen dazu folgen später.

Biologisch inspirierte Versuche, die Natur durch Algorithmen zu imitieren, sind nicht erst ein Thema der letzten Jahre. In den Bereichen neuronale Netze oder genetische Algorithmen konnte man von Zeit zu Zeit durchaus ansehnliche Erfolge verbuchen. Nichtsdestotrotz muss man eingestehen, dass man von der Komplexität natürlicher biologischer Systeme noch weit entfernt ist, und es ist durchaus vorstellbar, dass den Selbst-Eigenschaften ein ähnliches Schicksal bevorsteht.

3 ANSÄTZE UND METHODEN

3.1 Botenstoffe

Die Natur bedient sich spezieller Mittel, um selbstorganisierendes Verhalten zu erreichen, etwa des *Pheromons*. Die autonomen Einheiten (wie Ameisen, Termiten) platzieren diese flüchtige Duftstoffe und werden wiederum durch diese gesteuert. Schwarmintelligenz und Selbstorganisation in Multi-Agenten-Systemen beruhen derzeit hauptsächlich auf der Modellierung dieses Vorgangs der Stigmergie, der indirekten Kommunikation durch die Umgebung mit Hilfe von Botenstoffen.

Babaoglu et al. [16] haben eine Sammlung von „Entwurfsmustern“ („Design patterns“) für biologisch inspirierte Prinzipien und Basisprozesse in verteilten Systemen vorgeschlagen. Außer Stigmergie nennen sie noch *Diffusion* (Austausch durch Konzentrationsgefälle, vergleichbar mit Erosion und Nivellierung) auch als Bestandteil von Reaktions-Diffusions-Systemen, *Replikation* (epidemische Vermehrung) und *Chemotaxis* (Beeinflussung der Bewegung durch Stoffkonzentrationsgradienten). Im Wesentlichen beziehen sich diese Prozesse auf die Modellierung chemischer Duft- und Boten-Stoffe, also auf Pheromone und deren Wirkung. Dies bestätigt unsere Beobachtung, dass Pheromone das Hauptprinzip bei derzeitigen Versuchen sind, Selbstorganisation nachzuahmen, besonders im Bereich der Multi-Agenten-Systeme.

Während bei Insektenstaaten die Steuerung durch global wirksame, aber flüchtige Duftstoffe und Pheromone erfolgt, gibt es beim menschlichen Körper neben dem vegetativen Nervensystem das endokrine System, welches durch körpereigene globale Botenstoffe, wie Hormone, die Nachrichten von einem Organ zum anderen oder von einem Gewebe zum anderen transportieren. Neben den Pheromonen können daher auch Hormone als Modell für Selbstorganisation durch biochemische Botenstoffe dienen [17]. Im Gegensatz zu Pheromonen haben Hormone keine Wirkung auf die Systemumgebung, sondern wirken nur innerhalb des Systems, dort aber global. Sie sind Botenstoffe, die nicht sofort wirken, dafür aber länger anhalten.

3.2 Autonome Agenten

Autonome Agenten, insbesondere Softwareagenten, standen in ihren Varianten – den intelligenten und den mobilen Agenten – im Mittelpunkt des Interesses seit Mitte der 1990er Jahre bis in die letzten Jahre hinein [18, 19]. Das Konzept sieht, trotz vielerlei Auslegungen, die Autonomie der Agenten, also die *Selbständigkeit*, als unverzichtbaren Kernpunkt solcher Systeme vor.

Das Agentenmodell findet Verwendung in unterschiedlichen Schichten von IT-Systemen, so etwa schon auf Betriebssystemebene, meist jedoch auf der Ebene der Middleware oder auch als Anwendung zur Unterstützung des Anwenders. Ziel des Agentenentwurfs ist, dass Agenten selbständig Aktionen durchführen, um ihre inneren „Pläne“ im Auftrage des Anwenders zu verwirklichen. Der menschliche Anwender wird aus dem Entscheidungs- und Steuerungskreis ausgeklammert, so weit es möglich ist.

Der frühe Zuspruch zu dieser Technologie flachte mit der Zeit ab, da sich erwies, dass die Gestaltung von Anwendungen mit autonomer Entscheidungskompetenz nicht trivial ist. Mehr noch: Viele interessante Anwendungsfälle konnten von anderen Technologien behandelt werden, die keine radikale Abkehr von bewährten Entwurfsmustern erforderten. Insofern kann man die Agententechnologie als ersten Schritt zur Realisierung von autonomen Systemen mit „Selbst“-Eigenschaften sehen – und möglicherweise auch als Testfall für die nun angestrebte breite Nutzung dieser Eigenschaften in der Software.

3.3 „Selbst“-Eigenschaften

Mit den verwendeten Ansätzen und dem Dickicht der vielen „Selbst“-Begriffe konnte die Informatik bisher nicht an den Erfolg der Natur anknüpfen, was die Verwirklichung von Systemen mit „Selbst“-Eigenschaften angeht. Natürliche Systeme sind hier unübertroffen. Sie weisen oft ein ganzes Bündel von inhärenten „Selbst“-Eigenschaften aus, während man bei künstlichen Systemen fast jede einzeln „von Hand“ spezifizieren, programmieren und sicherstellen muss.

Hier ist deutlich der Einfluss einer unvollständigen Theorie zu spüren. Organisation völlig ohne Organisator zu verwirklichen ist häufig eine schwer durchschaubare Angelegenheit. Dabei kann es sich entweder um theoretische Grenzen handeln, die aus prinzipiellen Gründen nicht überschritten werden können, oder unser Erkenntnisfortschritt ist bislang einfach nicht groß genug. Deswegen kommen in der Regel explizite Manager-Elemente oder autonome Verwalter, die viel mit Agenten gemeinsam haben, zum Einsatz.

Eine anderer Grund mag sein, dass der derzeitige Ansatz, einzelne „Selbst“-Eigenschaften zu verwirklichen, nicht allgemein und systematisch genug ist, um die Art und Weise der Systeme von Grund auf zu ändern. Natürliche Systeme sind von Grund auf „selbstsüchtig“ – Dawkins hat die anschauliche Metapher des egoistischen Gens geprägt – und die Evolution selektiert auf natürliche Weise die Organismen, die sich selbst am besten schützen, anpassen, regenerieren, erhalten und reproduzieren können. Wesentlich systematischer wäre es deshalb, sich nicht an einzelnen „Selbst“-Eigenschaften zu versuchen, sondern Egoismus, also „Selbstsucht“ als wesentlichen Bestandteil des System-Fundaments von Beginn an mit einzubauen.

Alle (wünschenswerten) Aktivitäten, die das System „selbst“ verursacht, für die es selbst verantwortlich ist, kann man als „Selbst“-Eigenschaft bezeichnen. Die Menge all dieser Begriffe wird – in Anlehnung an eine Schreibweise für reguläre Ausdrücke – gerne auch als „Self-* properties“ oder entsprechend

„Selbst-*-Eigenschaften“ bezeichnet. Die Palette reicht von selbstschaffend bis zu selbsterhaltend, selbstmodifizierend und selbstverändernd.

Die beiden am häufigsten in wissenschaftlichen Artikeln genannten Varianten sind Selbstorganisation und Selbstverwaltung (oder Selbstmanagement), beides sehr allgemeine Begriffe, die eine Vielzahl von anderen „Selbst“-Eigenschaften umfassen. Der Begriff Selbstorganisation nimmt mit all den zusätzlichen Bedeutungen, die am Anfang genannt worden sind, eine klare Sonderstellung ein und ist gleichzeitig zusammen mit der Selbstverwaltung der allgemeinste. An speziellen „Selbst“-Eigenschaften finden sich des Weiteren die Begriffe Selbst-Heilung, -Konfiguration, -Optimierung, -Schutz, -Inspektion, -Reparatur, -Verjüngung, -Regeneration, -Einstellung, -Optimierung, -Modifikation, -Wartung, -Regulierung, -Beobachtung, -Diagnose, -Test, -beschreibend, -erklärend, -Kenntnis, -Bereitstellung und -Aktualisierung. Die ungefähren Häufigkeiten dieser Begriffe sind in Abb. 2 dargestellt.

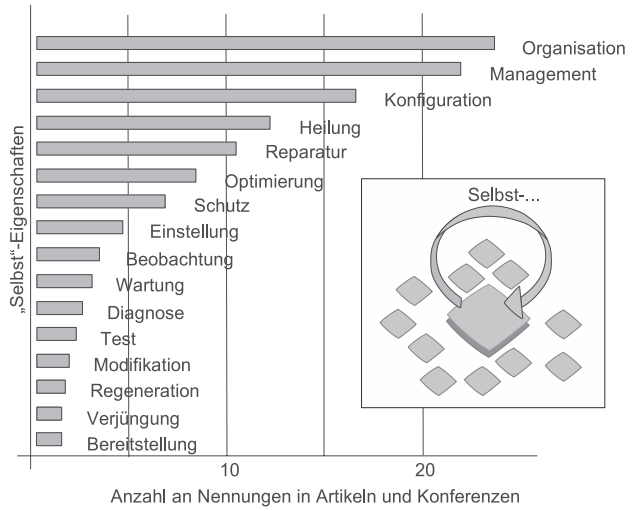


Abb. 2 Häufigkeit von „Selbst“-Eigenschaften [27, 28, 29, 30].

Es gibt zahlreiche Möglichkeiten, diese Begriffe zu klassifizieren oder zu kategorisieren, etwa nach Struktur-, Aktivitäts- oder Adaptions-bezogenen Eigenschaften. Fast alle „Selbst“-Eigenschaften haben jedoch die folgenden Aspekte gemeinsam:

- **Aktivitäten**: ein dynamisches Verfahren, eine Aktion, einen Prozess oder einen Vorgang
- **Autonomie**: das Recht für das System, über eine Reaktion zu entscheiden und sich selbst zu ändern
- **Feste Rollenaufeilung** in aktive und passive Elemente: Manager-Elemente überwachen und steuern die Vorgänge in „gemanagten“ Elementen
- **Modell**: (zumindest teilweise) Kenntnis des eigenen Systems, Reflexion

3.4 Autonomic Computing

Die Initiative *Autonomic Computing* wurde 2001 von IBM ins Leben gerufen. Hintergrund des Begriffs ist das *Autonomic Nervous System*, das *vegetative Nervensystem* [20]. Ziel ist, das Netz- und Systemmanagement in ähnlicher Weise wie das biologische Vorbild „im Hintergrund“ agieren zu lassen, ohne dass explizite Handlungen erforderlich wären.

Natürliche Systeme besitzen oft eine ganze Reihe von „Selbst“-Eigenschaften – im Gegensatz zu künstlichen Systemen. Die Vision des Autonomic Computing fasst deshalb die häufigsten und vielleicht wichtigsten vier „Selbst“-Eigenschaften in der IT zusammen:

- **Selbstkonfiguration**: Automatische Konfiguration von Komponenten und Systemen gemäß abstrakter Vorgaben.
- **Selbstoptimierung**: Komponenten können ihre eigene Leistung überwachen und wissen Mittel, diese zu verbessern.
- **Selbstschutz**: Früherkennung von Angriffen und eigenständige Gegenmaßnahmen.
- **Selbstheilung**: Das System erkennt innere Fehler und ist in der Lage, diese zu beheben (Hard- und Software)

Diese Prinzipien finden sich, wie Abb. 2 zeigt, nach Organisation und Management tatsächlich als häufigste Begriffe in Fachartikeln wieder.

Doch diese Festlegung stößt bisweilen auf Kritik, die den Begriff von IBM vereinnahmt sieht, wobei es weitere interessante Aspekte des vegetativen Nervensystems zu untersuchen und auf seine Tauglichkeit im IT-Netz- und Systemmanagement zu prüfen gilt. Insbesondere könnten, neben den funktionalen, gerade die infrastrukturellen Eigenschaften von Interesse sein: Es ist nicht das *Was*, sondern das *Wie*, das interessante neue Wege weisen kann. So sind nicht zuletzt die Kommunikationswege im Körper ein mögliches Vorbild zur Unterstützung selbstorganisierender Strukturen.

Die eigentliche Funktion, nämlich das Verlagern der Managementfunktion in den Hintergrund, das Herauslösen des Menschen aus dem Regelkreis, muss sich jedoch nicht zwangsläufig am Prinzip des Nervensystems anlehnen, sondern kann auch Analogien zur Ausbreitung von Pheromonen oder Hormonen nutzen.

Autonomic Computing nennt explizit das Vorhandensein einer Hierarchie von zentralen Steuerungskomponenten (Central managers). Insofern handelt es sich um keinen dezentralen Ansatz im eigentlichen Sinne. Andererseits sind dezentrale Regelkreise nicht unproblematisch: Selbst wenn sie einzeln das System in einen stabilen Teilzustand führen würden, müssen sie zusammengenommen nicht unbedingt das Gesamtsystem in einen stabilen Gesamtzustand führen [21].

Kephart berichtet in einem aktuellen Artikel über die Herausforderungen des Autonomic Computing [22], dass im Prinzip kein großer Unterschied zwischen Autonomen Managern und Agenten besteht. Insofern könnte die Agentenforschung eine Fülle an nützlichen Erfahrungen bieten, auch in Bezug auf Stärken und Schwächen.

3.5 Proactive Computing

Autonomic Computing besteht im Wesentlichen aus der Selbstverwaltung von Systemen, um den menschlichen Administrator weit gehend zu entlasten. In IBMs Vision soll der Mensch aus dem Regelkreis heraustreten und zahlreiche solcher eigenständigen Kreise sollen das Gesamtsystem in Gang halten. Diese Idee des „Human-out-of-the-loop“ ist auch Kernpunkt von *Proactive Computing*, einer Initiative von David Tennenhouse und Intel [23]. Sie ist quasi das Gegenstück von Intel zu IBMs Initiative des Autonomic Computing. Die Grenzen zu Ubiquitous Computing und Autonomic Computing sind fließend.

3.6 Organic Computing

Die *Organic-Computing*-Initiative [24] ist die deutsche Antwort auf die Vision des Autonomic Computing. Dabei liegt der Schwerpunkt stärker auf Hardware und rekonfigurierbaren Schaltkreisen. Nach dem biologischen Vorbild sieht das Organic Computing das Gesamtsystem als Zusammensetzung von Organen, die sich möglicherweise aus noch kleineren Bestandteilen, den Zellen, zusammensetzen. Außerdem versucht man, organische Computersysteme im Sinne von lebenden und lebensähnlichen Systemen zu schaffen, die vergleichbare Robustheit, Flexibilität und „Selbst“-Eigenschaften aufweisen.

3.7 Recovery-Oriented Computing (ROC)

Recovery-Oriented Computing ist ein Forschungsprojekt der Universitäten UC Berkeley und Stanford für selbstheilende, fehlertolerante und hochverfügbare Systeme [25].

Die These lautet: Störungen, Fehler und Ausfälle muss man erwarten – weil sie unvermeidlich sind. In der Regel ist eine schnelle Diagnose und Wiederherstellung möglich, indem das betroffene Teilsystem neu gestartet wird.

ROC formuliert seine Grundprinzipien so:

- *Micro-Reboot*: Minimiere die Auswirkung von Ausfällen. Jedes Modul (wie EJBs) kann unabhängig von anderen neu gestartet werden
- *Quick comeback*: Maximiere die Wahrscheinlichkeit einer schnellen Wiederherstellung des Systems

3.8 Epidemic Computing

Im *Epidemic Computing* versucht man sich das Prinzip der Kettenreaktionen, Infektionskrankheiten und Epidemien zunutze zu machen [26]. Dabei infiziert jeder Knoten seine Nachbarknoten, indem jeder Knoten von Zeit zu Zeit Informationen an eine Anzahl (zufällig ausgewählter) Nachbarknoten sendet. Eine Kettenreaktion oder Epidemie ist ein robuster und skalierbarer Prozess, der sich selbst aufrechterhält.

Mit Hilfe von Epidemic Computing kann man skalierbare und fehlertolerante Replikation und Aktualisierung von Informationen und Daten erreichen, Fehlerdetektoren konstruieren und Informationen zuverlässig verbreiten. Es ist daher vergleichbar mit kontrolliertem Fluten („Flooding“).

4 ANWENDUNGEN IN DER IT

Betrachten wir nun einige konkrete Anwendungen, in denen es hauptsächlich um die Umsetzung spezieller „Selbst“-Eigenschaften geht. Wie man auf Workshops wie dem Seminar im Herbst 2004 in Dagstuhl [27] sowie der SELF-STAR-Konferenz 2004 in Bologna [28, 29] und dem SelfMan-Workshop 2005 in Nizza [30] erkennen kann, haben die „Selbst“-Eigenschaften ihren systemtheoretischen Diskussionsrahmen längst verlassen und Eingang in aktuelle IT-Projekte auf akademischer wie auch industrieller Seite gefunden.

Wiederum zeigt sich, dass die Konzepte der *Selbstorganisation* und der *Selbstverwaltung* unter den Arbeiten am häufigsten auftauchen. „Selbst“-Eigenschaften finden sich in der Informatik-Forschung in zahlreichen Gebieten, von denen wir vier exemplarisch vorstellen:

- Neue Netzstrukturen und Netzdienste: World-Wide Web, Sensornetze, Adhoc-Netze
- Sicherheitstechnik: Anwendung des Organic Computing als künstliches Immunsystem
- Robuste Software und Protokolle mit Selbstheilung
- Simulationen und andere Anleihen aus der Biologie

4.1 World-Wide Web

Selbstorganisation war bereits vor 10 Jahren ein Thema. In [31] wird eine Selbstorganisation des World-Wide Web (WWW) anhand des Surfverhaltens vorgestellt. Ähnlich den Vorgängen im Gehirn sollten sich im WWW entlang der Verweise Strukturen ausbilden – die Verweise spielen also die Rolle der Verbindungen zwischen Nervenzellen, und je häufiger der Link benutzt wird, umso „stärker“ wird die Verbindung zwischen der verlassenen und der neu betretenen Seite. Ziel ist das Entstehen eines adaptiven, assoziativen Netzes, eines „weltweiten Hirns“.

Auf dieser Basis sollten sich thematische Komplexe herausbilden, die dem Anwender einen leichteren Zugang zu verwandten Themen (wie durch Assoziation) ermöglichen. Allerdings ist dieser Ansatz, wie die Autoren anführen, wesentlich von technischen Neuerungen in den verwendeten Protokollen abhängig; die Verwendung der Verweise insbesondere in HTML erweist sich für derart „wachsende“ Verbindungen als zu statisch.

Eine andere Form der Selbstorganisation im WWW findet sich in der populären Wiki-Bewegung, die in der *Wikipedia* ein weithin bekanntes und äußerst erfolgreiches Projekt vorweisen kann. Hier zeigen sich stigmergische Einflüsse: Gerade weil sich viele Menschen mit der Wikipedia beschäftigen und ihnen die Möglichkeit verliehen ist, in geordneter Weise zu einem größeren Ganzen beizutragen, lockt dies weitere Menschen an, ebenfalls ihre „Spuren“ zu hinterlassen.

4.2 Overlay-Netze

Netze, die auf der bestehenden Netztopologie zu einem speziellen (Anwendungs-)Zweck erzeugt werden, nennt man *Overlay-Netze*. Es sind im Prinzip logische Netze, die über die physischen Netze gelegt werden.

- *Peer-to-Peer-Netze* sind die prominentesten Beispiele; hier entstehen zwischen den Knoten Bekanntschaftsbeziehungen; manche Knoten spielen spezielle Rollen (zum Beispiel Indexserver).
- Webringe sind Verkettungen von Webseiten zu speziellen Themen
- Virtual Private Networks (VPNs) finden ihren häufigsten Einsatz bei der Kommunikation mit einem entfernten Netz, wobei der eigene Knoten als Teil des gesicherten Netzes Zugriff auf interne Ressourcen erhält
- Spezielle Anwendungen, wie Multi- oder Mobile-Agenten-Systeme, bilden ebenfalls solche Netze.

Obgleich P2P-Netze für ihre Anwendung zum Dateitausch – und dabei vornehmlich unlizenzierter Kopien von Mediendaten – bekannt wurden, ist ihre Bedeutung viel weitgehender. Eine der besonders interessanten Eigenschaften ist, dass die Ressourcen der verfügbaren Knoten wesentlich besser genutzt

werden. Neben der Verteilung von Daten kann das Ziel auch die Verteilung von Rechenkapazität sein.

Anders als in der vertrauten Klient-Server-Welt, in der ein zentraler Rechner alle Anfragen verarbeitet und Ergebnisse zurückliefert, finden sich in den Peer-to-Peer-Netzen (P2P-Netze) im Prinzip keine hervorgehobenen Knoten – ausgenommen zur Hierarchisierung des Netzes, um Anfragerouten zu vereinfachen.

Die Einrichtung solcher logischer Netze ist mit administrativem Aufwand verbunden, sowohl für die Einrichtung als auch für den Betrieb. Bislang können Netzknoten nur jene Dienste anbieten, die explizit installiert sind – die Funktionalität ist bislang nicht darauf ausgerichtet, frei umkonfiguriert werden zu können. Wünschenswert wäre es, wenn sich bei Bedarf solche Netze *selbst konfigurieren* könnten. Die Systeme erzeugen so selbständig virtuelle Netze, je nach Bedarf. Dabei replizieren sie insbesondere auch Dienste und stellen sie an verschiedenen Punkten zur Verfügung.

Selbstkonfiguration und *Selbstanpassung* hilft bei der Optimierung des Netzes für spezielle Aufgaben. Rechner mit geringer Belastung können automatisch als Indexserver für eine P2P-Netz eingerichtet werden. Ressourcen können bei Bedarf repliziert werden. „Active Networking“ und „Active Nodes“ sind weitere, wichtige Schlagwörter. Netzknoten bekommen zusätzliche Funktionen, die sich so gestalten lassen, dass der Knoten sein Verhalten ändern kann. Als Router kann er dynamisch auf spezielle Verkehrsmuster angepasst werden. Bei Videoübertragungen können Router mit dem Absender in Verbindung treten und eine Justierung der Bitrate verhandeln.

Im Zusammenspiel der prinzipiell gleichberechtigten Knoten ergeben sich Probleme der Koordinierung sowie der Nachrichtenleitung, die neuartige Ansätze erfordern: Einerseits entstehen neue Schwierigkeiten, wie etwa das Einfügen und Entfernen von Teilnehmern sowie die Kommunikation und das Finden von Ressourcen. Die für das Management von P2P-Netzen gefundenen Erkenntnisse ermöglichen andererseits neue Strategien für das weite Feld der eingebetteten oder allgegenwärtigen Systeme (Embedded Systems oder Ubiquitous Systems).

Die Robustheit solcher serverlosen Netze – es fehlt eben ein solcher zentraler Ausfallpunkt – ist ein attraktives Ziel, das im Angesicht immer wieder auftretender Störungen in den heute verwendeten Netzen an Bedeutung gewinnt.

4.3 Sensornetze und mobile Ad-hoc-Netze

Während bei Peer-to-Peer-Netzen bestehende IP-Netze mit einer übergeordneten Struktur versehen werden, findet man bei Sensor- und Ad-hoc-Netzen eine umgekehrte Situation vor: Hier möchte man zunächst eine Netzinfrastruktur erzeugen, die sich so wie ein gewöhnliches IP-Netz verhält, also die beliebige Adressierung von Knoten und Weiterleitung der Daten bietet.

Sensornetze bestehen typischerweise aus einer Vielzahl von miteinander vernetzten Sensoren, die zwar mehr als nur reine Messsonden darstellen, aber im Vergleich zu üblichen PCs, einschließlich der Handgeräte (wie PDAs), über sehr begrenzte Ressourcen verfügen – sowohl in Hinblick auf die Rechenleistung, also auch auf die Stromversorgung, und damit verbunden auch die Sendeleistung. Ferner kann der Zugang zu den Sensoren schwierig oder unmöglich sein – beispielsweise, wenn diese im Fahrbahnbelag vergossen werden oder die Sensoren in der Außenwelt verstreut werden. Ein zentraler Kommunikationspunkt erscheint somit schon technisch unmöglich.

Um solche Netze nach bisherigen Methoden in Einsatz zu bringen, wäre ein erheblicher Konfigurationsaufwand erforderlich, wobei man bestimmte Knoten als Router auszeichnen müsste, welche die Aufgaben der Weiterleitung von Daten an die von ihnen aus erreichbaren Knoten erledigen.

Sind bei Sensornetzen die Knappheit der Ressourcen in den Knoten das Hauptproblem, so rückt die Mobilität der Knoten bei MANETs in den Mittelpunkt. MANETs [32] sind so genannte drahtlose mobile Ad-hoc-Netze, welche für ihren Aufbau Selbstorganisation erfordern, um beliebige, nichtdeterminierte Netztopologien aufbauen zu können, wie Abb. 3 zeigt. Sie erfordern spezielle Vorkehrungen, um den Netzbetrieb robust zu gestalten. Das besondere Interesse an diesem Gebiet wird nicht zuletzt durch die große Menge unterschiedlicher Protokollansätze belegt [33].

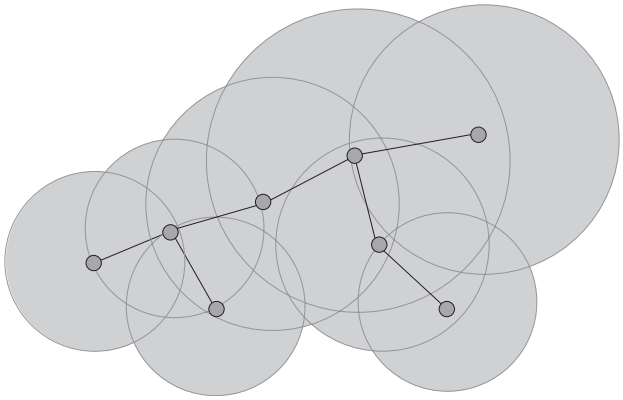


Abb. 3 Ad-hoc-Netz

„Selbst“-Eigenschaften spielen in solchen Szenarien eine vielfältige Rolle. Der Aufbau einer geeigneten Netztopologie (in welcher die Erreichbarkeit der Knoten gewährleistet ist), ist ein Ausdruck der *Selbstorganisationsfähigkeit* dieser Netze. Nach dem Aufbau ist die Erhaltung des Betriebs, wenn Knoten ausfallen, außer Reichweite geraten oder hinzugefügt werden, ebenso wenig von einem externen Administrator zu bewältigen wie der Aufbau. Es ist also wichtig, dass solche Netze über eine *Selbstverwaltung* verfügen, wobei solche Teilaufgaben wie die Wiederherstellung gestörter Routen auch als *Selbstheilung* bezeichnet werden kann.

4.4 Sicherheitstechnik

Biologische Immunsysteme werden gerne als Musterbeispiel einer dezentral organisierten, flexiblen und schlagkräftigen Abwehr gegen Eindringlinge herangezogen. Kämpft in einem bisherigen Sicherheitssystem der Sicherheitsbeauftragte, unterstützt durch seine Sammlung von Kontrollprogrammen, von zentraler Stelle gegen Übergriffe, so scheint es der (biologische) Körper zu schaffen, seine Immunantwort ohne direkte Einwirkung einer zentralen Steuerung zu organisieren.

Das Immunsystem verfügt über Sensoren, die lokale Aktivitäten auslösen, indem sie Botenstoffe aussenden. Diese Botenstoffe werden über die Blutbahn verteilt und führen damit zu einer Aktivierung des gesamten Abwehrsystems.

Über den Blutkreislauf ausgebreitet treiben die Botenstoffe nicht nur zu einem, sondern möglicherweise mehreren Zielen. Nur spezielle Zellen (mit den passenden Rezeptoren) reagieren auf das Vorhandensein dieser Stoffe. In der Informatik kann man solche Mechanismen mit Multicast oder Publish/Subscribe-Verfahren vergleichen.

Organic Computing versteht sich als Analogie zu den biologischen Vorgängen im Körper. In [34] wird demonstriert, wie die Prinzipien des Organic Computing unmittelbar auf das Gebiet der Erkennung und Abwehr von Eindringlingen in Netzen übertragen wird. Organic Computing zielt insbesondere auf „Selbst“-Eigenschaften wie die *Selbstheilung* und auch die *Selbstanpassung* ab.

Für ein Netzwerk könnte dies bedeuten, dass das Netz selbständig Teile abkoppelt, in denen Sicherheitsverletzungen stattfanden. Man spricht hier von einem *künstlichen Immunsystem.* Das natürliche Immunsystem erkennt unbekannte Eindringlinge und Fremdkörper; diese Fähigkeit möchte man nachahmen und nutzen [35]. Für eine verteilte Abwehr von Angriffen auf ein Netz ergeben sich aus dem biologischen Vorbild einige Analogien:

– Intrusion-Detection-Systeme (IDS) agieren als Sensoren.
– Im Falle des Einbruchs an einer Stelle des Firmennetzes melden diese ihre Beobachtungen und charakteristische Muster des Angriffs an verschiedene Stellen im Netz, insbesondere an Router und Firewalls.
– Auch hier werden die Erkenntnisse der IDS an alle Interessenten im Netz propagiert – ganz ähnlich der Blutbahn.

Gerade bei der Abwehr von Viren und Würmern ergibt sich das Problem, Schadcode zu erkennen, ohne vorher ein genaues Erkennungsmuster zu besitzen. Letztlich ist es erforderlich, schädlichen Code in ähnlicher Weise zu erkennen, wie das Immunsystem Eindringlinge bemerkt. IDS arbeiten hier bereits mit vergleichbaren Konzepten und fahnden etwa nach verdächtigem Absuchen offener Ports.

Ob es sich bei einem Prozess um einen viralen Prozess handelt, lässt sich also vielleicht an seinem Verhalten erkennen, etwa auch, wenn er wichtige Dateien zu überschreiben versucht. Code an sich noch vor der Ausführung als schädlich zu erkennen, ist im Allgemeinen nur dann möglich, wenn bestimmte Teile des Codes als „bösartig“ bekannt sind und entdeckt werden. Auch hier bietet die Biologie mit der „negativen Selektion“ Anleihen: Es wird eine große Vielfalt von Detektoren produziert, von denen alle, die auf „gute“ Muster reagieren, eliminiert werden. Was übrig bleibt, sind demnach Detektoren für fremde, potenziell gefährliche Muster.

Organic Computing bietet eine interessante Möglichkeit, den DDoS-Angriffen (Distributed Denial-of-service), eine der gefürchtetsten Gefahren für den Betrieb von Netzdiensten, zu begegnen. Spürt ein System eine wachsende Last auf, kann es sich *selbst replizieren* und auf einem Knoten laufen, welcher der aktuellen Anfragewelle nicht ausgesetzt ist. Auf diese Weise kann zumindest für all jene, welche diese zweite Lokation kennen, der Betrieb aufrechterhalten werden.

Wenn es sich nicht gerade um einen DDoS-Angriff, sondern lediglich um eine Phase sehr hoher Belastung handelt, erstellt das System eine Kopie seiner Konfiguration auf anderen verfügbaren Rechnerknoten. Gegebenenfalls kann auch der Datenbestand repliziert werden. Nützlich ist in diesem Falle eine zusätzliche Indirektionsschicht im Netzwerk. Anstatt mit einer Adresse einen bestimmten Rechner zu adressieren, wird der Zugriff auf einen dieser nun redundanten Knoten gelenkt. IPv6 könnte hier wertvolle Hilfe mit dem ANYCAST-Konzept leisten.

4.5 Bewahrung der Dienstgüte

Auch der sehr biologisch anmutende Begriff der *Selbstverjüngung* hat mittlerweile in IT-Konzepte Eingang gefunden, gerade durch die Arbeiten im Bereich des Recovery-Oriented Computing.

Oft wird beobachtet, wie Anwendungen mit fortschreitender Laufzeit merklich in ihrer Leistung nachlassen oder deutlich mehr Fehler produzieren. Diese sind jedoch meist nicht genau lokalisierbar – sei es, dass es keine Möglichkeit gibt, die Programme zu analysieren, sei es aufgrund der wachsenden Komplexität der Systeme an sich.

Nimmt die Qualität der Diensterbringung (etwa die Performanz) aus diesen nicht offensichtlichen Gründen ab, kann man von *Alterung* der Anwendung sprechen. Entsprechend bezeichnet die *Verjüngung* Maßnahmen, um die erwarteten Dienstgüte wieder zu erreichen, beispielsweise durch das Ersetzen oder durch einen Neustart von Komponenten oder des Gesamtsystems [36].

Interessant wäre es, diesen Vorgang zu automatisieren, das System also zur *Selbstverjüngung* zu ermächtigen. Voraussetzung ist eine laufende *Selbstüberwachung*, aber auch ein Modell, welches Voraussagen über die Reaktionen des Systems bei solchen Eingriffen erlaubt, also prädiktive Fähigkeiten. Man könnte hier von Reflexion, von *Selbstkenntnis* sprechen. Prinzipiell muss das System Informationen über die Wirkung seiner Komponenten besitzen.

4.6 Robuste Anwendungen und Protokolle

Zurzeit arbeiten wir allerorten mit Anwendungen, die in gewissem Sinne brüchig sind [37]: Jeder Fehler im Programm führt in bestimmten Situationen zum Fehlverhalten des gesamten Programms. Beim bisherigen linearen Programmiermodell ist jede Programmzeile ein potentieller Ausfallpunkt.

Solche Fehler aufzudecken ist sehr aufwändig, denn es muss sich nicht immer um einfache, isolierbare, fehlerhafte Ausdrücke oder Befehle handeln, zumal mehr und mehr Verantwortung vom Programmierer auf Interpreter oder Compiler verlagert wird. Auch wenn Compiler und Interpreter die gröbsten Fehler selbst entdecken, treten dann noch immer Fehler in der Programmlogik auf, letztlich aufgrund der Komplexität der Software.

Nimmt man wieder die Natur zum Vergleich, stellt man fest, dass nicht jeder Schaden zum Versagen führt, und mehr noch – Schäden werden selbst repariert. Daher ist man bestrebt, Programme so zu schreiben, dass sie fehlertolerant sind. Dies ist besonders für kommunikative Software von Relevanz: Wie können Protokolle so robust gestaltet werden, dass Fehler nicht nur erkannt, sondern auch behoben werden?

Anstatt jede Programmzeile für sich stehen zu lassen, werden austauschbare Module bereitgestellt, die bei Fehlverhalten einfach ersetzt werden können. Ist ein Programm (lokal oder ver-

teilt) in dieser Weise gestaltet, so ist es zur *Selbstheilung* befähigt.

Die Software sollte in der Lage sein, von mehreren möglichen Ausführungspfaden den richtigen auszuwählen. Das „chemische Rechnen“ (Chemical Computing) weist interessante Wege: So „reagieren“ Protokolleinheiten mit bestimmten Teilen des Programms, werden zu neuen „Verbindungen“, die ihrerseits wieder Reaktionspartner finden. Fällt ein Reaktionspartner aus, ist gewöhnlich ein anderer, gleichartiger Partner vorhanden, welcher die Aufgabe übernimmt.

In [38] wird ein solches Verfahren für einen einfachen Fall beschrieben: Tatsächlich funktioniert das Protokoll selbst nach Ausfall von Teilen noch immer korrekt, und man kann sogar nachvollziehen, welche Teile ausfielen.

4.7 Simulationen und weitere biologische Anleihen

Simulationen biologischer oder soziologischer Phänomene sind natürlich ein unmittelbar passender Einsatzort für computerimplementierte „Selbst“-Eigenschaften und Emergenz. So lassen sich recht eindrucksvoll Schwärme von Fischen oder Insekten sowie Futter suchende Ameisen simulieren und visualisieren; sie legen dabei entsprechend eine künstliche Pheromonspur an, nutzen also Konzepte der *Stigmergie* [39, 40]. Für diese Simulation gibt es nützliche Anwendungen. So können auf diese Weise kürzeste Pfade in einem Netz bestimmt oder die Lastverteilung effizient organisiert werden [41].

Genetische Programmierung (GP) [42] ist eine besonders augenfällige Übertragung biologischer Prinzipien: Programme evolvieren, ähnlich den biologischen Arten, gemäß ihrer Eigenschaft, in den Problemraum „besser zu passen“. Das System zeigt also *Selbst-Entwicklung*. Anfang der 1990er Jahre noch kaum über die Lösung einfacher Probleme hinausragend, hat die GP mittlerweile einige frappierende Ergebnisse geliefert, die man eigentlich von einem Menschen erwartet. Man spricht daher bereits von einer „Erfindungsmaschine“ und „Automatischer Programmierung“ [43, 44].

Die GP ist ihrerseits eine Weiterentwicklung der *evolutionären Algorithmen*, die sich als bewährtes Mittel für Optimierungsprobleme etablierten und so auch zur Selbstoptimierung eingesetzt werden. Erstmals eingesetzt von John H. Holland um 1970, gibt es inzwischen eine reichhaltige, einschlägige Literatur [45, 46].

5 EIN KRITISCHER BLICK

Betrachtet man Netzwerkstrukturen an sich, so ist die Situation noch übersichtlich. Im Wesentlichen gibt es zwei wichtige Phänomene – skalenfreie Netze und die sogenannte Kleine-Welt-Eigenschaft – die häufig vorkommen, die gewünschte Eigenschaften wie Skalierbarkeit und Robustheit haben und noch dazu mit einfachen Regeln leicht zu erzeugen sind. Geht man jedoch auf Anwendungen über, auf welche die genannten Initiativen wie Autonomic und Proactive Computing sowie die einzelnen, zahlreichen „Selbst“-Eigenschaften ausgerichtet sind, ist die Situation wesentlich komplizierter.

Viele dieser Initiativen überlappen und imitieren sich mehr oder weniger stark. Die Vielfalt der Forschungsansätze und Methoden deutet zudem darauf hin, dass der eigentliche Kern des Problems, Selbstorganisation in verteilten Systemen zu verwirklichen, noch nicht gelöst ist. Es ist relativ leicht, von einer neuen Ära zu reden [47] oder eine neue x-Computing-Initiative zu erfinden wie Amorphous Computing, Organic Computing, Ubiquitous Computing und so weiter – doch sie nutzbringend auszugestalten ist wesentlich schwieriger.

Möglicherweise ist der Preis für neuartige selbstorganisierende Systeme, dass wir einen Teil der Vorhersagbarkeit, Bestimmtheit und Kontrolle abgeben müssen, wie Kelly argumentiert [48], sei es an autonome Agenten oder evolutionäre Algorithmen. Das bedeutet, dass man nicht alle Dinge exakt vorhersagen, im Voraus bestimmen und bis ins Detail festlegen kann.

Ein weiteres zentrales Dilemma bei der Umsetzung der Prinzipien, Phänomene und Konzepte von selbstorganisierenden Systemen in der Natur ist folgendes: Entweder ist das System robust, anpassungsfähig und skalierbar, oder es ist zuverlässig, vorhersagbar, verständlich und bis ins Detail kontrollierbar. Beides zusammen erscheint schwierig: Ein zentrales Element, welches das autonome System als Ganzes regelt, steht der Skalierbarkeit von verteilten Systemen im Wege. Viele einzelne autonome Elemente, die miteinander agieren, können das Verhalten das Gesamtsystems wiederum durch emergente und unerwartete Eigenschaften unvorhersehbar machen. Das bedeutet, dass für verteilte Systeme, die gerade Selbstverwaltung und Selbstorganisation erst besonders notwendig machen, nur dezentrale autonome Elemente in Frage kommen, die wiederum andere Eigenschaften wie Vorhersagbarkeit und Zuverlässigkeit erschweren.

Prinzipien wie Emergenz, Schwarmbildung und Schwarmintelligenz sind inzwischen recht gut bekannt und in Simulationen wiederzufinden, bei der praktischen Nutzung spielen sie aber so gut wie keine Rolle. Sie werden oft in Multi-Agenten-Systemen studiert, in denen man sich bevorzugt auf die Implementierung einzelner „Selbst“-Eigenschaften konzentriert, wie der Abschnitt über Anwendungen in der IT gezeigt hat. Einer vollständigen Lösung des Problems „Selbstorganisation in verteilten Systemen“ kam man bislang damit offenbar nicht entscheidend näher.

Einzelne „Selbst“-Eigenschaften sind nur ein kleiner Schritt in Richtung „selbstorganisierende Systeme“. Stattdessen wäre es wesentlich systematischer, die Art und Weise zu verändern, wie die Systeme aufgebaut sind. Am weitesten wagt sich IBM mit seiner Vision des Autonomic Computing [20], die gleich vier zentrale „Selbst“-Eigenschaften umfasst, und mit „Proactive Computing“ und „Organic Computing“ zwei Mitstreiter gefunden hat. Sie stellen die Entwickler vor große Herausforderungen, und bei genauem Hinsehen stößt man auf die gleichen Probleme wie bei Multi-Agenten-Systemen [22].

Ist der systematische Einsatz von Multi-Agenten-Systemen eine Lösung? Je nach Art des Agentensystems lassen sich die Prinzipien selbstorganisierender Systeme wie Emergenz, Schwarmbildung oder Schwarmintelligenz wesentlich besser und wirkungsvoller als bei traditionellen objektorientierten Softwaresystemen einsetzen. Aber auch diese Systeme haben ihre Vor- und Nachteile. Sie sind zwar in der Regel besser skalierbar, robuster und anpassungsfähiger als reine objektorientierte Systeme, aber Zuverlässigkeit, Vorhersagbarkeit und Verständlichkeit können abhängig vom Grad der Autonomie und der Art der Systeme mehr oder weniger stark eingeschränkt sein.

6 ZUSAMMENFASSUNG UND AUSBLICK

Wir haben gesehen, dass sich gute Gründe für Selbst-Organisation und Selbst-Management überall dort finden, wo verteilte Systemen besonders ausgeprägt in Erscheinung treten. Leider ist der Begriff Selbst-Organisation sehr allgemein und kann zu Missverständnissen führen, weil die genaue Bedeutung vom jeweiligen Kontext abhängt. In der Informatik ist man an selbstorganisierenden Systemen interessiert, die robust, skalierbar und anpassungsfähig sind. Wie diese am besten zu realisieren sind, ist noch unklar. Es gibt eine Reihe von Konzepten und Prinzipien in biologischen Systemen, deren Übertragung auf IT-Systeme nicht unproblematisch ist. Wir haben die Vor- und Nachteile beschrieben, sowie anschließend anhand konkreter Projekte den Stand der Technik erläutert.

Alle Projekte und Initiativen im Umfeld der Selbst-Organisation nehmen sich zum Ziel, letztlich den Menschen seiner Verantwortung für eine dauerhaft fehlerfreie Funktion von Infrastruktur und Anwendungen zu entbinden. Der Wunsch besteht, dass sich die Arbeit „wie von selbst“ erledigen solle. Doch man gibt auch Eingriffsmöglichkeiten ab – denn man kann nur schwer gleichzeitig die Regelung komplexer Vorgänge einem System überlassen, zugleich aber über jeden Schritt informiert bleiben.

Die Vielzahl an Arbeiten im Umfeld der „Selbst“-Eigenschaften und selbstorganisierenden Systeme lassen vermuten, dass fortwährend wesentliche Erkenntnisse gewonnen werden – aber tatsächliche Einsichten und grundsätzliche Realisierungen sind noch immer rar. Offenbar müssen an die Stelle von philosophischen Überlegungen und Simulationen sehr viel mehr praxisbezogene Experimente und Szenarien treten.

Ob daher der Implementierung einzelner „Selbst“-Eigenschaften oder eher systematischen Ansätzen zur Erzeugung von selbstorganisierenden Systemen mit emergenten Phänomenen eine breite Akzeptanz zuteil wird, muss die Zukunft zeigen.

LITERATUR

[1] Truszkowski et al: NASA’s Swarm Missions – The Challenge of Building Autonomous Software, IT Professional, Vol. 6, No. 5, September/October (2004) 47-52.

[2] Barroso, Dean, Hölzle: Web Search for a Planet – The Google Cluster Architecture, IEEE Computer, März-April 2003; S. 22-280.

[3] Weiser: Some Computer Science Problems in Ubiquitous Computing, Communications of the ACM, Vol. 36, No. 7, July (1993) 75-84.

[4] Chlamtac, Conti, and Liu: Mobile Ad Hoc Networking – Imperatives and Challenges, Elsevier Ad Hoc Networks Journal, Vol. 1, No. 1, July (2003) 13-64.

[5] Bonabeau et al.: Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press (1999).

[6] Nachtigall: Bionik, Springer, 2002.

[7] Camazine et al.: Self-Organization in Biological Systems, Princeton University Press (2003).

[8] Whitesides, Boncheva: Beyond molecules: Self-assembly of mesoscopic and macroscopic components, PNAS, Vol. 99, No. 8, April 16 (2002) 4769-4774.

[9] Bak: How Nature works, Springer, 1996.

[10] Jantsch: Die Selbstorganisation des Universums, DTV, 1982.

[11] Prehofer, Bettstetter: Self-Organization in Communication Networks – Principles and Design Paradigms, IEEE Communications Magazine, Vol. 43, No. 7, July (2005) 78-85.

[12] Johnson: Emergence, Scribner, 2001.

[13] Holland: Emergence, Oxford University Press, 1998.

[14] Watts: Six Degrees, W.W. Norton & Company, 2003.

[15] Barabasi: Linked, Perseus, 2002.

[16] Babaoglu et al.: Design patterns from biology for distributed computing. Erscheint im Konferenzband zu „European Conference on Complex Systems“, November 2005.

[17] Shen et al.: Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots, 17: S. 93-105, 2004.

[18] Webseite zu AgentLink III: European Coordination Action for Agent-based Computing. http://www.agentlink.org/

[19] Webseite der FIPA: Foundation for intelligent physical agents. http://www.fipa.org/

[20] Kephart, Chess: The Vision of Autonomic Computing, IEEE Computer Magazine, 2003.

[21] Herrmann, Mühl, Geihs: Self-Management: The Solution to Complexity or Just Another Problem? IEEE Distributed Systems Online, Band 6, Nr. 1, 2005.

[22] Kephart: Research Challenges of Autonomic Computing, Konferenzband zur ICSE 05 (2005) 15-22.

[23] Tennenhouse: Proactive computing, Communications of the ACM, Volume 43, Issue 5, May (2000) 43-50.

[24] VDE/ITG/GI-Positionspapier zum Organic Computing, verfügbar unter http://www.gi-ev.de/

[25] Candea, Brown et al.: Recovery-Oriented Computing: Building Multitier Dependability, IEEE Computer, November (2004), S. 60-67.

[26] Vogels, van Renesse, Birman: The Power of Epidemics: Robust Communication for Large-Scale Distributed Systems, ACM SIGCOMM Computer Communication Review Vol. 33, No. 1, January (2003) 131-135.

[27] Service Management and Self-organization in IP-based networks. Seminar auf Schloss Dagstuhl, 03.10.-06.10.2004. Webseite: http://drops.dagstuhl.de/portals/04411/

[28] SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems, 31. Mai – 2. Juni 2004 in Bologna. http://www.cs.unibo.it/self-star/

[29] Self-star Properties in Complex Information Systems: Conceptual and Practical Foundations; Hrsg: Özalp Babaoglu, Márk Jelasity, Alberto Montresor et al.; Springer-Verlag, LNCS 3460/2005; ISBN: 3-540-26009-9.

[30] SelfMan 2005 IFIP/IEEE International Workshop on Self-Managed Systems & Services, http://madyne.loria.fr/selfman2005

[31] Algorithms for the self-organisation of distributed, multi-user networks. Possible application to the future World Wide Web (1996): http://pespmc1.vub.ac.be/Papers/AlgorithmsWeb.pdf

[32] IETF MANET working group: http://www.ietf.org/html.charters/manet-charter.html

[33] Wikipedia (englisch): Ad-hoc protocol list. http://en.wikipedia.org/wiki/Ad_hoc_protocol_list

[34] Bio-inspired mechanisms for efficient and adaptive network security. http://drops.dagstuhl.de/opus/volltexte/2005/87

[35] CARIS 2004: 3rd International Conference on Artificial Immune Systems, Catania, Sizilien. Springer-Verlag, LNCS 3239, 2004.

[36] Malek, Salfner, Hoffmann: Self Rejuvenation – an Effective Way to High Availability. SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems, Bertinoro, Italien, Juni 2004; http://www.cs.unibo.it/self-star/papers/malek.pdf

[37] Griffiths: Organisation and Emergence. PCPro-Magazin (2001). Online unter http://www.adit.co.uk/html/organisation_and_emergence.html

[38] Tschudin, Yamamoto: A Metabolic Approach to Protocol Resilience. First IFIP-Workshop on Autonomic Communication (WAC 2004), Oktober 2004, Berlin. http://cn.cs.unibas.ch/people/cft/doc/2004-protocol-resilience.pdf

[39] Webportal „Stigmergic systems“: http://www.stigmergicsystems.com/

[40] Wilensky (1999) NetLogo Ants Model, http://ccl.northwestern.edu/netlogo/models/Ants

[41] Schoonderwoerd, Holland, Bruten: Ant-like agents for load balancing in telecommunication networks. Konferenzband zu First International Conference on Autonomous Agents 1997, ACM Press. http://dsp.jpl.nasa.gov/members/payman/swarm/schoon97-icaa.pdf

[42] Koza: Genetic Programming – On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[43] 36 Human-Competitive Results Produced by Genetic Programming. Webseite der Forschungsgruppe zur Genetischen Programmierung: http://www.genetic-programming.com/humancompetitive.html

[44] Koza, Keane, Streeter: Evolving inventions, Scientific American, February 2003.

[45] Weicker: Evolutionäre Algorithmen, Teubner, 2002.

[46] Gerdes u.a.: Evolutionäre Algorithmen, Vieweg, 2004.

[47] Metropolis, Rota (eds.): A New Era in Computation, The MIT Press, 1993.

[48] Kelly: Das Ende der Kontrolle. Die biologische Wende in Wirtschaft, Technik und Gesellschaft, Bollmann Verlag, 1997.