Organic Computing: Zusammenfassung

Alexander Osiik
alexander.osiik@student.uni-luebeck.de
Organic Computing (WS 2019/2020)
Institute of Computer Engineering, University of Lübeck

2. Februar 2020

1 Einführung

Computersysteme wrden immer komplexer. Die Netzwerke von kleinen, eingebetteten Systemen werden immer größer und komplexer. Es stellt sich die Frage, wie man unter solchen Bedingungen die Komplexität von

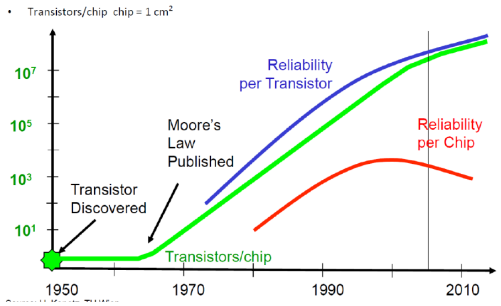
- Design
- Implementierung
- Management
- Adaptation etc.

betrachtet und verbessert.

In Organic Computing möchte man deswegen selbst **organisierte Systeme** entwickeln, die automatisch höherlevelige **Ziele** erfüllen und sich an die **Wünsche des Nutzers** anpassen. Dabei orientiert man sich oft an **lebenden Organismen**.

1.1 Motivation

Man möchte bei komplexen Systemen herausfinden, wie sie **funktionieren**, wie man sie **designed** und wie man sie **überwacht**.
Moore'sches Postulat



Bei komplexen Systemen wie autonomen Fahrzeugen stellen sich viele Fragen.

- Wie **überwacht** man die Komplexität eines solchen Systems?
- Wie stellt man sicher, dass es keine **ungewollte Interaktion** zwischen zwei Systemen gibt?
- **WER** kontrolliert das Auto?
- Wer wird bei Fehlern zur **Rechenschaft** gezogen?

Man möchte sich die Entwicklung komplexer Systeme dort abgucken, wo es schon erfolgreich war (zB. Gehirn).

Komplexe Systeme entwickeln sich meistens durch **natürliche Selektion / Evolution**, oder durch tiefgründige **Forschung**, wo mehr und mehr komplexe selbst-organisierte Systeme gefunden werden können.

1.2 Emergenz

Unter Emergenz versteht man die spontane **Entwicklung von Systemcharakteristiken**, die sich durch Interaktion mehrerer Entitäten entwickeln. Diese emergenten Eigenschaften können nicht bei einer **isolierten** Entität erklärt werden.

- Interaktion zwischen mehreren Entitäten
- **keine** zentrale Kontrollinstanz
- resultierende Patterns sind nicht vorher explizit programmiert

Die Herausforderungen liegen darin, das emergente Verhalten **integrativ** zu verstehen (micro→macro) oder das Verhalten individueller Elemente herauszufinden (macro→micro).
Beispiele: Shockwave Traffic Jam, V-Formation Vögel

1.3 Traditionelle Perspektive

"Deterministic Centralized **(DC)** Mindset"

- **D**: Relevanz von Randomisierung und Rauschen nicht berücksichtigt
- **C**: Pattern wird durch Koordinator generiert

Eine falsche Annahme ist, dass bei systematischen oder systemweiten Effekte nur ein einziger Aktor betrachtet wird.

1.4 Organic Computing

Bei Organic Computing betrachtet man das Zusammenspiel von (eingebetteten) Systemen mit bestimmten Eigenschaften, wie **viele Entitäten**, **Mobilität**, **Vernetzung** und spontane lokale Interaktionen **(Emergenz)**.

Was wir dafür brauchen sind viele, selbst-organisierte Entitäten, **robuste** Verhalten, flexible Verhalten, **Adaption** an dynamische Umweltfeatures und **Vertrauen** sowie **Kontrolle**.

Kann man ein selbst-organisierenden und adaptierenden Mechanismus erstellen, ohne alle mögliche Zustände zu kennen?

1.4.1 Meta-Design

- Der Ingenieur macht Gesetze, kein Micromanagement
- Fuzzy-Platzierung statt rigide
- Adaptieren und selbst regulieren statt Kontrolle und Redesign

1.5 Verhalten von Maschinen

Das Verhalten von Maschinen ist ein kompliziertes wissenschaftliches Feld, sobald es genauso schwer zu verstehen ist wie ein biologisches System.

1.5.1 Verhalten

Verhalten ist, wie sich ein Lebewesen in bestimmten Situationen oder als Reaktion auf einen bestimmten Stimulus verhält.

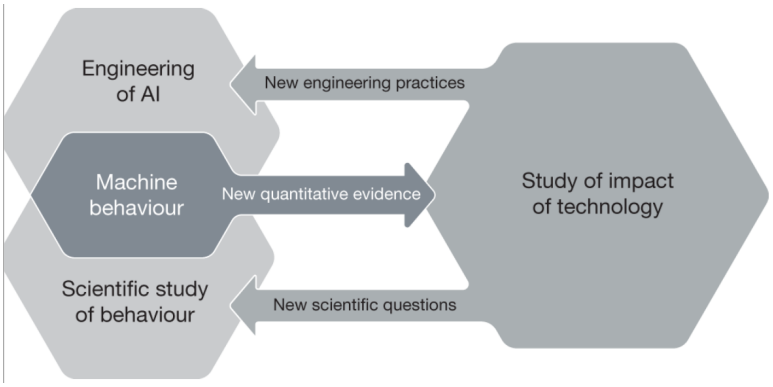
Es gibt sozial geprägtes Verhalten, Verhaltenswissenschaften (Psychologie, Sozialwissenschaften, Neuralwissenschaften)

1.5.2 Tierverhalten (Ethologie)

Wissenschaftliche und objektive Forschung von Tierverhalten unter natürlichen Voraussetzungen. Verhalten wird als adaptives evolutionäres Extra behandelt.

1.6 Komplexität und Opazität von Algorithmen

Architektur und Training können sehr simpel sein, jedoch ist das Resultat sehr komplex, eher eine **black box**. Exakte funktionale Prozesse sind **hart zu interpretieren**, sogar von Wissenschaftlern, die diese Algorithmen erstellen! Sourcecode und Modell sind nur bedingt prädiktive Kraft.



Man soll zunächst **individuelles**, **kollektives** und dann **hybrides Mensch-Maschine** Verhalten studieren!

2 Von Bewegung zu Intelligenz

Organic Computing ist Überwachung von komplexen technischen Systemen, und wie man diese Systeme **selbst-x**(adaptivm optimierend, reparierend etc.) macht. Man will also, dass diese System **intelligent** sind; sich in nicht antizipierten Situationen clever verhalten.
Intelligente Entscheidungsfindung ist das angemessene Verhalten in neuen Situationen, wobei man die Handlungsalternativen basierend auf den eigenen Präferenzen aussucht.

Doch was ist die **Grundlage** der Intelligenz, und wie können wir sie **finden** oder **integrieren**?

2.1 Wichtigkeit der Bewegung

Die Wahrnehmung der eigenen Bewegung und die anderer Entitäten ist Grundlage der Intelligenz. Es ist eine wichtige Kompetenz, die über das Überleben in der Wildnis entscheidet. Was sich bewegt macht einen **lebendigen** Eindruck.

2.1.1 Randomisierte Bewegung

Die **randomisierte** Bewegung, bei der ein System sich in irregulärer Weise bewegt.

$$\sigma_s = \sqrt{\langle s_N^2 \rangle} = \sqrt{N} \quad \text{Bewegung nach Münzwurfmodell}$$

Beim Random Walk wächst die Distanz zum Ursprung proportional mit der Zeit. Die unterliegende Wahrscheinlichkeitsdichte ähnelt der eines Diffusionsprozesses.
Brown'sche Bewegung: Zufällige Bewegung von Partikel in einer Flüssigkeit abhängig von Temperatur Bewegung.
Drunkard's Walk: Man startet eine Bewegung an Position c_0, jeder Schritt c_i hat die **Länge L**. Nach N Schritten ist die Distanz $\sqrt{N}L$ erreicht.
Weitere Variationen: Brown'sche Bewegung mit Drift:

$$\frac{\triangle x}{\triangle t} = \mu + \sigma \frac{\triangle W}{\triangle t}$$

2.1.2 Ziel-orientierte Bewegung

Beispiel: Bakterielle Chemotaxis
Hierbei haben Bakterien zwei Bewegungstypen:
Geradlinige Bewegung (Rotation CCW), und **Taumeln**(Rotation CW). Der **Bias** ist hier die zeitliche Wahrnehmung, ob die Situation sich verbessert (weiterschwimmen) oder nicht (Taumeln). Die Modellierung erfolgt anhande von

- Intrazellulären Modell
- Zellulären Model
- Populationsmodel (interzellulär)

2.2 Interaktion

Es gibt grundsätzlich zwei Formen der Interaktion

- Explizit: Es wird ein beabsichtigtes Signal gesendet
- Implizit: Physischer Kontakt, Kommunikation über die Umgebung

Die Signale können **Visuell**(Glühen, Farben), **Auditiv**(Heulen, Rufen), **Taktil**(Anfassen) und **Chemisch**(Pheromone, olfagorisches) sein

Bewegung nach Münzwurfmodell Kommunikation/Population braucht man denn? Wieviel Neuronen braucht man für Intelligenz? Wieviele Leute für eine gute Party?
Beispiel: Das Drei Körper Problem aus der Physik (Modellierung von 3 frei beweglichen abhängigen Partikeln)

2.2.1 Simple Interaktion

N Partikel, die sich zufällig bewegen, Gruppen detektieren, stoppen, und eine gewisse Zeit warten. Wie aggregiert man alle N Partikel in ein Cluster?
Ansatz: In großen Clustern länger bleiben.

2.2.2 Kollaboration

Bei zwei Agenten hat jeder die Wahl zwischen zwei möglichen Optionen:
"**Cooperate**" oder "**Defect**"

	C	**D**
C	R, R	S, T
D	T, S	P, P

Das **Iterated Prisoner's Dilemma** untersucht das obige Verhalten nach mehreren Runden. Hierbei ist die Frage, welche Strategie am schlausten ist (Keine Perfekte Information vorhanden, da Entscheidungsbaum simultan!):

- Revenge!
- Sofort Heimzahlen, aber 2.Chance geben
- Immer Nett sein
- Random

2.2.3 Evolution

Idee: Wir benutzen künstliche Evolution für IPD um Strategien zu entwickeln und zu schauen, welche Strategie überlebt. Die getesteten Stratefien sind

- Konditionale Kooperator
- betrügerische Überläufer
- Purer Überläufer

Man beobachtet jeweils eine zyklische Dominanz.

2.2.4 Grün-Bart-Effekt

Der Grün-Bart-Effekt ist ein Gedankenexperiment in verwendete Evolutionsbiologie selektiv zu erklären Altruismus unter den Individuen einer Spezies. Zwei sich erkennende Individuen benehmen sich **vetternwirtschaftlich**.

Dazu gehört

- Fakultatives Helfen: Nur GB helfen
- Obligatorisches Helfen: Immer helfen, aber nur GB profitiert
- Fakultatives Schaden: Immer den ¬GB schaden
- Obligatorisches Schaden: Immer schaden, aber GB immun

2.3 Emergente Taxis

Kann ein Kollektiv von simplen Agenten zusammen etwas erreichen, was ein einzelner nicht schafft?
taxis: Bewegung als Antwort auf einen Stimulus

Bei der **Emergent Phototaxis** möchte man

1. Die Kollision mit anderen Robotern vermeiden
2. Die Kohärenz beibehalten
3. Roboter, die Kohärenz erlangen, bewegen sich random.

Fazit: Es funktioniert!

3 Kollektive Intelligenz

In Agentensystemen gibt es wenig bis **gar keine zentralisierte Kommunikation** oder Kontrolle. Es können nicht alle Agenten reden.

Exemplarische Design Probleme, die verteilte Computersysteme beinhalten, sind

- Kontrollsystem für Routing in einem Netzwerk
- Routing in einem Stromnetz
- Verkehrskontrolle

3.1 El Farol Bar

Das El Farol Bar Problem ist ein Problem aus der Spieltheorie. Jeden Donnerstag wollen die Bewohner in die El Farol Bar. Die Bar ist klein, und niemand hat Spaß, wenn es zu voll ist.
Wenn weniger als 60% gehen, haben alle mehr Spaß, als sie zu Hause gehabt hätten. Vice Versa.
Problem: Jeder entscheidet zur selben Zeit!

3.2 Weisheit der Menschenmenge

Idee: Man sammle Meinungen von einer Anzahl an Experten. Diese Meinungen werden verarbeitet (Mittel/Median). Die "**kollektive**" Lösung ist meistens **besser**, als die "**individuelle**".
Francis Galton, Plymouth 1906

Die Effektivität der Weisheit wird oft auf das **Gesetz der Großen Zahlen** reduziert, bei dem der Durchschnitt eines statistischen Experiments den Erwartungswert μ erreicht, je mehr Versuche gemacht werden.
Hier ist es jedoch wichtig, dass die **Unabhängigkeit** jeder Wertung vorliegt, und Experten sich nicht gegenseitig beeinflussen. Das **Mitteln** reduziert zufällige Fehler, aber nicht **systematische Fehler**.

3.3 Kollektive Entscheidungsfindung

3.3.1 Best-Of-n Problem

Man möchte aus n Alternativen die beste finden.

- jede Option hat die Qualität $\rho_i \in (0, 1]$
- Mehrheit $M \geq (1 - \delta)N$ wird gesucht, mit $0 < \delta << 0.5$

Wie lange braucht der Prozess zum konvergieren?
Wie oft wird die Mehrheit erreicht?

3.3.2 Ising Model

Populäres Modell, welches aus dem Ferromagnetismus kommt. Jede Zelle eines Grids hat dabei die Zustände $s = \pm 1$. Nachbarn wollen dabei im gleichen Zustand sein (Energieminimierung). Ein spontaner Wechsel ist abhängig vn der Temperatur möglich.

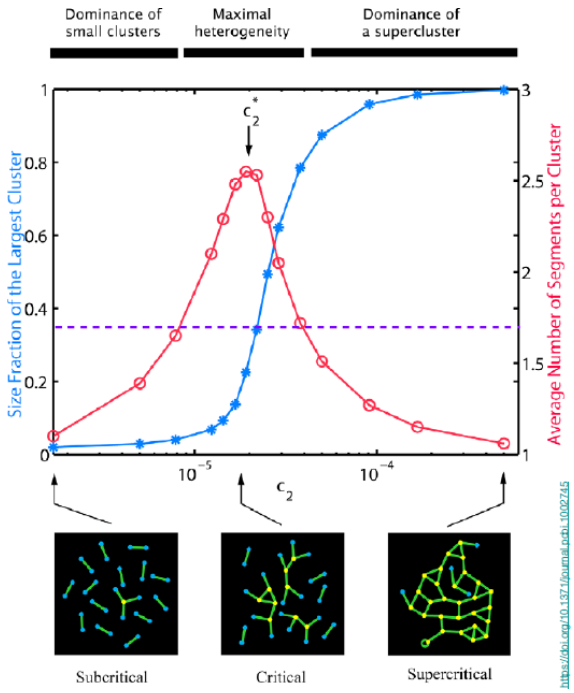
- Magnetisierung $M = \sum_i s_i$
- Energie $E = \sum_{\langle ij \rangle} s_i s_j$

Mit dem **Metropolis Algorithmus** wird pro Zeiteinheit ein Spin geflippt und geguckt, ob die Energie sich reduziert. Die Minimierung wird akzeptiert, wenn:

1. $\triangle E \leq 0$: Exploitation mit WS $= 1$
2. $\triangle E < 0$: Exploration mit Wahrscheinlichkeit $A = \exp(-\beta \triangle E)$, $\beta = \frac{1}{T}$

3.3.3 Perkolation (Versickerung)

Perkolation ist der Wasserfluss durch ein solides Substrat. **Perkolationstheorie** wird als Model für viele Szenarien verwendet, zb: Ausbreitung von **Epidemien**, **Waldbrände**, **Leitungsfähigkeit**...
Dabei beobachtet man ein interessantes, nicht-lineares Verhalten. Es gibt eine **kritische Wahrscheinlichkeit** p^*, wodurch ein kleines Cluster zu einem maximal großem Cluster mutiert.
Beispiel: Ein Feuer würde bis p^* ausgehen, danach jedoch immer durchbrennen.



3.3.4 Individuelle Entscheidungsfindung

Idee: Es gibt **Beweise** für eine der Alternativen, welche verarbeitet und **akkumuliert** werden bis eine **Schwelle** überschritten wird, nach der die **Entscheidung** getroffen wird.

3.4 Geschwindigkeit vs. Genauigkeit

Man kann schnell und ungenau sein, oder langsam und genau. Dies hängt nicht nur von Reaktionszeit ab!

Bei Kollektiven bekommt man meist Informationen durch seine Nachbarn. Beim **Ising-Modell** sind die Nachbarn fix, bei regulären Schwärmen bewegen sich die Nachbarn ständig! Die **lokale Stichprobe ist repräsentativ** für die gesamte Situation.

- Majority Rule (MR): Zähle die Meinungen der Nachbarn, bleib bei der Mehrheit (schneller)
- Voter Model (VM): Nimm einen zufälligen Nachbarn und übernimm seine Meinung (bei großen Schwärmen genauer)

4 Modellierung von OC-Systemen

Start: Man möchte komplexe Systeme überwachen.

- Was ist ein **System**?
- Was ist ein **komplexes System**?
- Wie können wir sie überwachen und **Performance** feststellen?

4.1 Technische Begriffe

4.1.1 System

Ein System ist eine Menge detaillierter Methoden, Prozeduren und Routinen, welche eine bestimmte **Aktivität** ausüben. Es ist eine organisierte **zweckgebundene Struktur**, die aus von einander abhängigen **Entitäten** abhängt. Diese Elemente beeinflussen sich gegenseitig und **erhalten die Existenz des Systems.**

Das **lineare System** ist dabei die simpelste Form eines Systems. Eine Änderung/Störung löst eine Antwort des Systems aus, welche proportional zur Eingabe ist.

$$y = Ax + B$$

Bei **nicht-lineare Systemen** ist die Systemantwort dementsprechend nicht proportional zur Eingabe, aber dennoch **vorhersehbar**.

$$y = \sin x$$

Nicht-lineare chaotische Systeme scheinen unvorhersehbar zu sein, obwohl deren Systemgleichungen deterministisch sind. Ein solches System ist stark von den **Anfangsbedingungen** abhängig.

$$x_{n+1} = rx_n(1 - x_n)$$

4.1.2 Komplex vs Kompliziert

Bei komplizierten Systemen ist die Lösungshierarchie **flach** (Evolution macht komplizierte Strukturen), während sie bei komplexen Systemen **vertikal**(Emergenz macht komplexe Strukturen) ist.

4.1.3 Systemperspektive

Betrachtung aller Verhalten des Systems als ein ganzes im Kontext seiner Umgebung ist die Systemperspektive. Bei der Systemperspektive soll das System bei weiterer Betrachtung nicht weiter reduziert werden.

4.1.4 Komplexität

Eigenschaft eines Systems, **alle** seine Eigenschaften nicht durch **einen** einzigen Formalismus adäquat beschreiben zu können.

4.1.5 Modellierung

Ein Modell ist eine vereinfachte Formalisierung der durch den Modellbauer verstandenen Welt (nicht zwingend realitätsgetreu). Für agenten-basierte Modelle modelliert man **Probleme**, **nicht Systeme**!

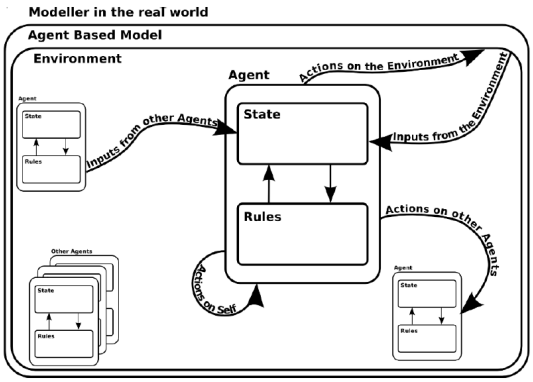
Beim **Top Down** Ansatz wird das System vom ganzen ins kleine modelliert. Voraussetzung dafür ist, dass das Verhalten bekannt ist, welches man reproduzieren möchte. Man braucht also ein gutes **Verständnis vom gesamten System** und wie die Komponenten **interagieren**.

Beim **Bottom Up** Ansatz wird das Verständnis vom System jedoch mehr gestärkt. Man versteht, wie relativ einfache Einheiten miteinander vernetzen, und wie ein **deterministischer Regelsatz** zur Entwicklung **komplexer Verhalten** führt.

Zu den agenten-basierten Ansätzen gehören **KI** (Agent ist autonome Einheit, welche Probleme löst), **Multi-Agenten Systeme** (Verteilte Kontrolle, kollektive Zielverfolgung) und **agenten-basierte Modellierung**.

4.2 Agenten

Ein Agent ist ein **eingekapselter** Computer, welcher in einer Umgebung ausgesetzt ist. Er kann durch **flexible, autonome Aktionen** in dieser Umgebung ein vorgegebenes Ziel verfolgen.



4.2.1 Zustand / State

Ein Zustand ist das Wissen über den Status der Umgebung. Dieses kann **öffentlich** oder **intern** sein. Zu jedem bestimmten Zeitpunkt ist der Zustand s als Element der Zustandsmenge $S = \{s_1, s_2, \dots\}$ (Intern: $I = \{i_1, i_2, \dots\}$) vorgegeben.

4.2.2 Regelnsatz

Neben Zuständen verfügt der Agent über einen Regelsatz, welche die interne Beschreibung der Prozesse darstellt. Dies können zB Transitionsregeln sein. Dabei gibt es die Varianten

- regelbasiert

- abhängig von mehreren konkurrierenden Zielen
- Heuristiken
- Machine Learning

4.2.3 Aktion

Eine Aktion hängt von anderen Agenten, internem Zustand (indirekte Umgebung) und dem Regelsatz ab. Eine Aktion beeinflusst dabei genau diese.

4.2.4 Umgebung

Eine Umgebung ist alles, was kein Agent ist, aber relevant für das Modell ist. Sie gibt **Informationen**, **Struktur** und **Stimuli**. Sie beeinflusst die Agenten, und wird von ihnen beeinflusst.

4.2.5 Zeit

Bei Agentensystemen betrachtet man in der Regel **diskrete Zeit**. Ein Zeitschritt ist dabei alles zwischen zwei Zeitpunkten, in diesem Zeitschritt funktioniert alles **simultan**. Agenten-basierte Systeme laufen **parallel** und **asynchron**, die Auswahl der Erstberechnungen muss also **randomisiert** erfolgen, um **Unabhängigkeit** zu schaffen.

4.2.6 Agententypen

- **Agenten mit Wahrnehmung (Reflex Agenten):**
  Sensordaten sind Wahrnehmung, aus der der Zustand der Umgebung bestimmt wird. Danach erfolgt eine Aktion.
- **Agenten mit internen Zuständen:**
  Agent hat internes Gedächtnis um den Zustand des Systems zu speichern. Der neue Zustand ist vom alten Zustand abhängig und der Wahrnehmung. Abhängig von **neuem** Zustand wird dann die Aktion ausgeführt.

4.3 Multi-Agenten-Systeme

Eine Menge von Agenten die in der gleichen Umgebung koexistieren. Dadurch können Aufgaben auf mehrere Agenten **aufgeteilt** werden, um die **Systemperformance** zu erhöhen, oder es können **komplexe Probleme** gelöst werden, die eine einzelner Agent nicht lösen kann.

Wie koordiniert man die Agenten durch Kommunikation und Interaktion?

Wichtige Eigenschaften sind:

- Die Agenten sind **autonom**
- Ihnen steht nur **lokale** Information zur Verfügung
- Das System ist **dezentralisiert**

- Zur Kollaboration und Koordination müssen Aktionen **geplant**, **mitgeteilt** und **verhandelt** werden.

4.4 Koordinationsmechanismen

Bei der **Korrelation** gibt es nur eine lose/zufällige Verbindung zwischen bestimmten Events. Es herrscht **keine Kausalität**.

Koordination ist eine Eigenschaft der Agentengruppe, die eine Aufgabe löst. Es ist ein **kausaler** Prozess, der aus Kommunikation der Agenten entsteht.

4.4.1 Kommunikationstypen

Die Kommunikation von Agenten kann durch **Kommunikationstopologie**(Zentral, Dezentral) und **Informationsfluss**(Direkt, Indirekt) kategorisiert werden.

- **Topologie:**
  Zentral: kein Informationsfluss zwischen Slaves
  Dezentral: Peer2Peer
- **Informationsfluss:**
  Direkt: Agent zu Agent, ohne Umgebung zu berücksichtigen
  Indirket: Durch die Umgebung (Constraints durch Master)

4.4.2 Typen von Koordinationsmechanismen

- **Markierung-basiert:** (Zeichen, Signal)
  Markierungen können von anderen Agenten observiert werden, andere Agenten können Tags beeinflussen oder hinzufügen (Vertrauen und Sicherheit)
- **Token-basiert:**(alles, was Agenten in einer Gruppe teilen können)
  Für jedes Token gibt es nur wenige Instanzen, Agenten mit Token haben **exklusiven** Zugriff. (Selbstschutz und Rollenverteilung)
- **Markt-basiert:** (Kaufen, Verkaufen)
  Agenten kommunizieren durch Transaktionen (Ressourcen Zuweisung)

Beispiel: Ameisen nutzen Pheromone zur Kommunikation. Pheromone ist die akkumulierte Erfahrung aller Ameisen. Diese "Information" kann **aggregiert** werden oder **verdampfen**. Jeder Agent kann auf die Information zugreifen und Information hinzufügen. **(Dezentralisiert, Indirekt, Markierung-basiert)**

4.4.3 Self X (OC in anderen Ländern)

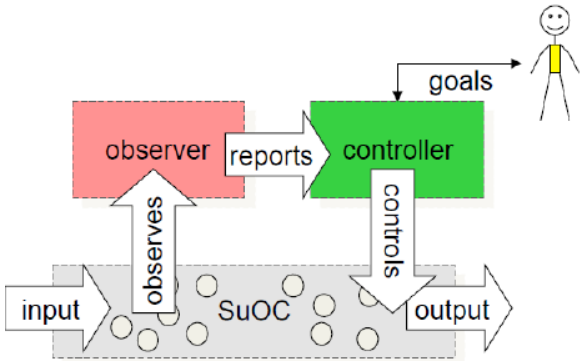
- **Self-Healing:** Das System kann sich von Fehlern erholen
- **Self-Protection:** Das System identifiziert und wehrt sich gegen Angriffe
- **Self-Optimizing:** Das System finden ein optimales Verhalten für ein Szenario
- **Self-Adaptive:** Das System passt sich an Veränderungen der Umwelt an

5 Architekturen

Man möchte das komplexe System, was man beobachtet, auch **beeinflussen**. Man möchte damit auch sicherstellen, dass das System innerhalb der **gegebenen Parameter** operiert.

5.1 Observer-Controller Architektur

5.1.1 System under Observation and Control (SuOC)



Start: Ein einzelnes System, bestehend aus einer Gruppe von autonomen Robotern, welches als komplett bezeichnet werden kann.
Input sind alle Informationen (**Umgebungseffekte**), die ein SuOC zum reagieren braucht, aber nicht beeinflussen kann.
Output sind alle Informationen (**Parameter**) die vom SuOC bestimmt werden und von außen zugänglich sind.

Ein SuOC muss dabei folgende Anforderungen erfüllen:

- **Ziele** erreichen und **Constraints** einhalten
- Verhalten von SuOC muss **beobachtbar** sein
- Performance muss **messbar** sein
- Muss **dynamische Kontrollfaktoren** besitzen

Der **Observer** (Beobachter) muss den aktuellen Systemzustand messen und evaluieren. Aus diesen Informationen muss er den nächsten Zustand vorhersagen können.

Der **Controller** "überwacht" den Prozess. Er **beeinflusst** das System so, dass die **Ziele erreicht** und die **Einschränkungen eingehalten** werden.
Er unterbricht dabei **unerwünschtes emergentes Verhalten** und **rekonfiguriert** das System, um der Entwicklung von unerwünschtem Verhalten entgegenzuwirken.

Möglichkeiten, das SuOC zu beeinflussen, sind:

- Verhalten von Agenten ändern (**Regelsatz**)

- Das **Netzwerk** modifizieren um das Agentenverhalten zu ändern
- Die **Anzahl** an Agenten(typen) erhöhen oder verringern
- Die **Umgebung** modifizieren

5.1.2 Observer Modelle

Der Controller kann dem Observer vorgeben, nach welchem **Observer-Modell** er handeln soll. Das Observer-Modell entscheidet darüber, welche **Features** überwacht werden solle, wie die **Daten analysiert** werden und wie der **zukünftige Zustand** vorhergesagt wird.

5.1.3 Observer

Der Aufbau des Observers ist wie folgt:

1. **Monitor:** Überwachung der ausgewählten **Parameter** zu vorgegebener **Sampling-Rate**. Diese werden in einem **Log-File** festgehalten, welche als **Basis für Vorhersagen** genutzt wird.
2. **Pre-Prozessor:** Bereitet die Daten für das Verarbeiten vor (Filterung und Aggregation)
3. **Data Analyzer:** Generiert den aktuellen Systemzustand (Parameter, entstehende Effekte). Die Methoden der Analyse hängen dabei vom gewählten Observer-Modell ab
4. **Predictor:** Vorhersage des zukünftigen Systemzustands. Erlaubt dem Controller, Entscheidungen nicht nur abhängig von der Vergangenheit zu treffen. Dadurch wird die Reaktionszeit reduziert (Proaktives Verhalten)
5. **Aggregator:** Kombiniert die oben genannten Daten zu "**Situationsparametern**", die dem Controller übergeben werden.

5.1.4 Controller

Der Aufbau des Controllers ist wie folgt:

1. **Mapping:** Implementiert Reaktion A_i auf bestimmte Situationsparameter C_i. Erlaubt auch **erlernte Reaktionen** für unbekannte Situationen
2. **Performance Evaluation:** Evaluation basiert auf Adaption von Selektoren. Abhängig davon wird das **Mapping** aktualisiert. **Online Learning** mit Historie von Aktions- und Situationsparametern
3. **Simulation Model:** Erlaubt das generieren und evaluieren neuer Regeln. Diese werden durch Modifikation anderer Regeln oder Neuschöpfen hinzugefügt, nachdem sie in der **Simulation erfolgreich** waren (**Offline Learning**). Bietet dadurch mehr Systemsicherheit.

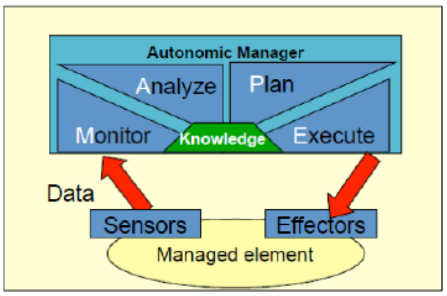
Die Organisation kann dabei **zentral**, **dezentral** oder **multi-level** erfolgen.

5.1.5 Beispiel: Aufzugskontrolle

Observer observiert en Bunching Effekt, Controller sorgt dafür, dass die Aufzüge äquidistant verlaufen.

5.2 Monitor-Analyze-Plan-Execute

Historie: Problem bei IBM. Rechnersysteme werden komplexer und die Administration wird komplizierter. Man benötigt ein Paradigmenwechsel, um weniger Leute einstellen zu müssen und Kunden zufrieden zu machen.



5.2.1 Autonomic Computing

Computer, die sich **selbst verwalten** können, wenn ihnen ein **high-level Ziel** gegeben wird. Ähnlich dem menschlichen Nervensystem sollen auch Computern oder Verteilten Systemen **vitale Funktionen** zur Verfügung stehen.
Siehe auch: **Self-CHOP Prinzipien**

5.2.2 Architektur

- **Autonome Systeme** sind ein Verbund von interagierenden autonomen Elementen
- **Autonome Elemente** besitzen Ressourcen und stellen Services zur Verfügung (für Nutzer oder andere AE). Sie interagieren mit anderen AE um Ziele zu erreichen. Sie haben einen oder mehrere zu verwaltende Elemente.
- Managed Elements (ähnlich wie SuOC) können ihren eigenen Kontrollzyklus haben. Beispiel hierfür wären Hardware- oder Softwareressourcen.

Der Autonome Manager ist für die Automation des IT Management Funktionalität verantwortlich. Er ist in der Regel für vier Aufgaben zuständig:

- **Überwachen** von ME
- **Analyse** von Daten
- **Planen** von Aktionen
- **Ausführen** von Aktionen

5.2.3 Monitor

Automatische **Kollektion**, **Aggregation**, **Korrelation** und **Filterung** von Informationen über die Managed Ressource. Die Daten werden gesammelt, bis ein Symptom entdeckt wird. Die beobachteten Symptome werden zur Analyse weitergereicht.

5.2.4 Analyse

Zustäandig für Einhaltung von Beschränkungen. Observiert und analysiert Symptome, lernt Systemverhalten und sagt zukünftige Zustände voraus.

5.2.5 Plan

Generiert und wählt Prozeduren aus die die Managed Ressource beeinflussen. Generiert einen Plan der von GuideLines abhängt.

5.2.6 Execute

Zeitplanung und Ausführung der geforderten Systemänderungen. Kann mehrere oder einen einzigen **Managed Ressource** beeinflussen.

5.2.7 Knowledge Source

Repositorium für Symptome, Guidelines, Änderungsanfragen etc. Kann von vielen AM aus zugegriffen werden.

5.3 Fazit

- "Wissen" ist über das O/C System verteilt
- MAPE: Sequenz von Operationen, O/C: Architekturkomponenten
- Bei O/C liegt der Fokus auf definierten Zielen
- MAPE hat keine expliziten **Offline Learning** Regeln
- Bei beiden gibt es keine Interaktion zwischen autonomen und organische Elementen

6 Schwarm Intelligenz (ACO)