Spezifikation

Cell Computing Model

Autor	Benjamin Hadorn
E-Mail	bhadorn@swissinfo.org
Ablage/Website	http://www.xatlantis.ch
Datum	13.12.05
Version	0.5.0

Inhaltsverzeichnis

1 Einleitung

Seit dem Beginn der Computer wurden immer wieder neue Technologien und Methoden entwickelt, um gewisse Probleme einfacher und strukturierter zu lösen. Mit der Maschinensprache wurden die ersten Programme für Computer geschrieben. Die darauf folgenden Hochsprachen waren einfacher zu lesen und anzuwenden, die Programmierung erfolgte aber nach dem gleichen Prinzip, nämlich linear (Prozedural). Es entstanden Probleme mit der Strukturierung von umfangreicherer Software. Die Lesbarkeit und dadurch die Wartung der Software waren schwierig und kompliziert. Mit der objektorientierten Programmierung (OO-Programming) wurde eine „neue“ Denkweise ins Leben gerufen. Die Modellierung der Software wurde an die reale Welt angepasst. Objekte aus der realen Welt werden abstrahiert und in die Entwicklung der Software miteinbezogen, sie werden also abgebildet. Diese Objekte können miteinander kommunizieren und Daten austauschen.

In den letzten Jahren ist der Umfang der Software und deren Anforderungen ständig gestiegen. Mit der weltweiten Vernetzung, dem Internet, wurde die Entwicklung von verteilten Systemen (zum Bsp. Web-Services, Java) stark gefördert. Die Objekte befinden sich nicht mehr nur auf einem Computer, sondern existieren dezentral und kommunizieren über das Netzwerk miteinander. Das Ziel ist, in einem Netzwerk von Computer verschiedene Dienste bereitzustellen, welche von anderen Computer oder Computersystemen gebraucht werden können.

Was die Zukunft bringt weiss niemand. Aber der heutige Trend in der Computerwelt zeigt, dass die Informationsmenge nicht geringer wird. Im Gegenteil. Die Verarbeitung der Informationen wird immer aufwendiger. Bilder können mit hoch genauerer Auflösung erstellt und gespeichert werden, Filme werden mit hoher Qualität über Netzwerke verteilt und umgewandelt. Bald wird das Fernsehen live und mehrkanalig auch über das Internet übertragen. Aber auch im Bereich der Geschäftswelt wird die Datenmenge grösser. Die Vernetzung der Gesellschaft spiegelt sich auch in den Systemen wieder, welche die Gesellschaft braucht, um ihre Aufgaben zu erledigen. Dazu werden in der Geschäftswelt immer kürzere Antwortzeiten verlangt. Das erfordert einerseits schnelle und anderseits flexible Systeme. Der Wunsch nach Systemen, die sich autonom regenerieren und sich an der Umgebung und der Aufgabe anpassen, wächst. Heute werden diese Systeme mit dem objektorientierten Ansatz entwickelt. Durch das Anwenden von Design Patterns und anderen Implementationstechniken wird versucht schneller und durchgängiger zu entwickeln. Die Entwicklungszeit solcher Systeme ist jedoch relativ lang und deshalb auch teurer. Die finanziellen Ressourcen der Käufer ist mehr oder weniger begrenzt, und so ist es auch verständlich, dass solche Systeme in der Geschäftswelt nicht gerade zu einer Euphorie führen.

2 Was ist Cell Computing Model?

Es stellt sich die Frage, ob nicht ein neues dem OO-Programming übergeordnetes Konzept geschaffen werden könnte, um die genannten Probleme und Anforderungen an zukünftige Systeme besser in den Griff zu bekommen. Wie könnte ein solches System aussehen?

Es gibt in der realen Welt einen riesigen Unterschied zu den heutigen Computersystemen: In der realen Welt laufen unendliche Ereignisse (Tasks) parallel ab, Arbeiten und Abläufe geschehen gleichzeitig. Es existieren unzählige Kreisläufe, die das System stabilisieren und ausbalancieren. In der Computerwelt stellt sich nun langsam die Frage, ob solche Verhaltensweisen nicht auch erwünscht wären. Die Datenmenge artet langsam in eine richtige Flut aus, das Verarbeiten wird dadurch nicht kürzer. Auch mit den immer leistungsfähigeren Mikroprozessoren wird einmal Schluss sein. Hier stösst die Technik heute schon an gewisse Grenzen. Doch die Vernetzung und Verarbeitung wird mit dem Wachstum unserer Gesellschaft stetig zunehmen, vielleicht sogar exponentiell. Dieses Dokument zeigt eine Möglichkeit auf, wie zukünftige Software geschrieben werden kann, nämlich mit den Erkenntnissen der Genforschung.

Die Natur könnte uns also auch hier wieder einmal als Vorbild dienen, um solche Systeme zu entwickeln. Das folgende Unterkapitel befasst sich kurz mit der realen Welt und welche Prinzipien zum Bau des Systems gebraucht werden können.

2.1 Die Welt der chaotischen Systeme

Die Erde besteht aus 3 Hauptkreisläufen, der Atmosphäre, den Meeresströmungen und der Magmaströmungen. Diese Kreisläufe sind chaotisch organisiert und beeinflussen sich gegenseitig. Sie geben dem System Erde die Stabilität, welche es schlussendlich ermöglicht, dass Lebewesen existieren können. Die Lebewesen ihrerseits helfen aber auch mit, das System zu steuern und zu stabilisieren. Sie sind in diese Kreisläufen integriert. Wird ein Kreislauf instabil, wird er durch andere Systemkomponenten wieder stabilisiert. Dabei ist das Prinzip der Evolution nicht irrelevant. Komponenten, die sich störend entwickeln, werden zerstört oder mit der Zeit verändert und an die Umgebung angepasst. Dieses Prinzip hat sich über mehrere Milliarden Jahre bewährt.

Beim Betrachten eines Lebewesens, wird klar, dass sich auch die internen Systeme eines jeden Lebewesens ähnlich verhaltet wie das ganze System

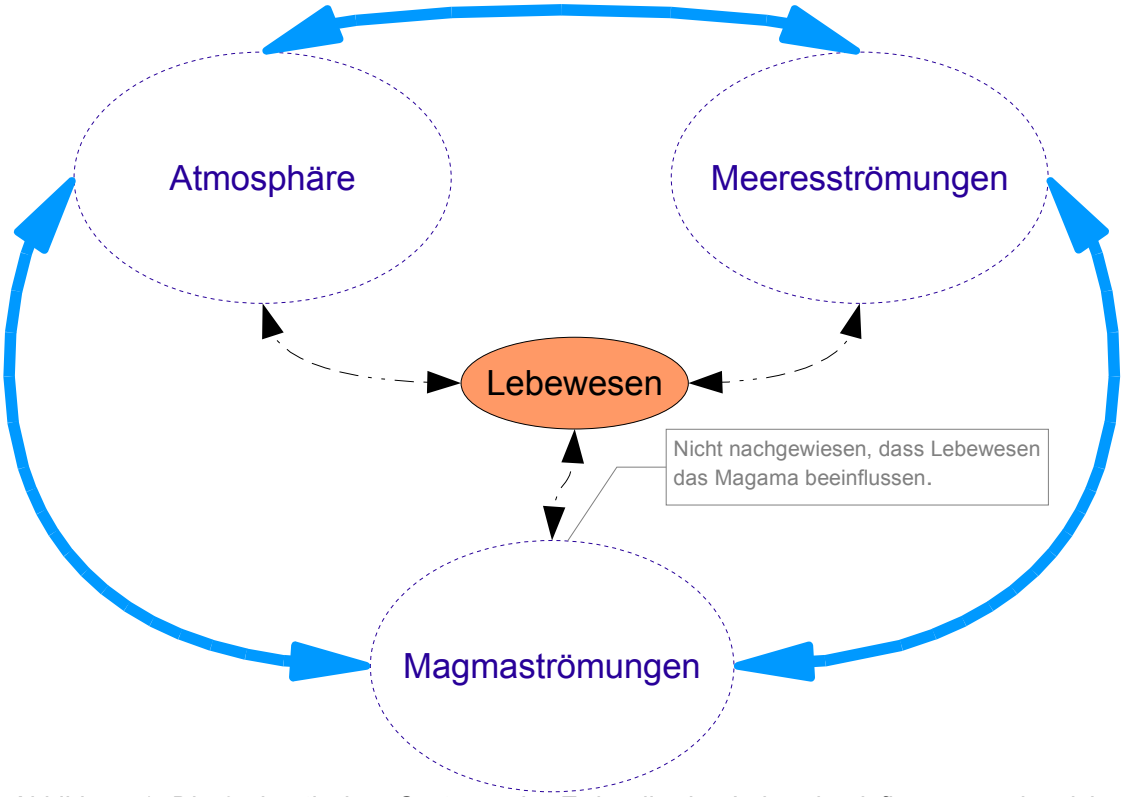

Abbildung 1: Die 3 chaotischen Systeme der Erde, die das Leben beeinflussen, und welche vom Leben beeinflusst werden.

Erde. Krankheiten werden bekämpft, das Immunsystem aufgebaut und weiter entwickelt.

2.1.1 Aufbau der Lebewesen

Die Lebewesen auf der Erde sind aus Zellen aufgebaut. Die Zellen stellen sozusagen kleine Systeme dar, die mit der Aussenwelt und ihren benachbarten Zellen interagieren. Diese Interaktion ist für die Zelle lebenswichtig. Sie können also nicht als eigenständig bezeichnet werden. Sie brauchen die Umgebung, um zu existieren. Aber auch die Umgebung ist von den Zellen abhängig. Die Zelle ist fest in einem oder mehreren der 3 Kreisläufe beteiligt.

Je nach Komplexität des Lebewesen gibt es unterschiedliche Zellenverbunde (Zellenhaufen). Die Zellenverbunde bestehen aus einem bestimmten Zellentyp, der dem Lebewesen einen bestimmten Dienst erweist.[1]

1 Wissen über die biologischen Zusammenhänge stammen aus [GEOK2] und [MSENC]

2.1.1.1 Stammzellen

Interessant ist das jedes Lebewesen aus einer einzigen Zelle entsteht. Diese Zelle teilt sich nun, es entsteht ein Zellenhaufen und schlussendlich das vollständige Lebewesen.

Stammzellen sind die Bausteine des Lebens (Urzellen). Sie können sich fast unbegrenzt teilen. Aus ihnen entwickeln sich alle anderen Zellen eines Lebewesens. Die ersten Zellen sind die embryonale Stammzellen. Aber auch nach der vollständigen Entwicklung des Lebewesens existieren Stammzellen, um den Körper zu heilen und zu reparieren.

2.1.1.2 Spezialisierte Zellen

Aus den Stammzellen entstehen spezialisierte Zellen. Diese nehmen nun eine bestimmte Aufgabe wahr. Sie sind die Bausteine der Organe (Haut, Leber etc.), der Nerven, Muskeln und des Bluts. Je nachdem, wie gross und umfangreich die einzelnen Aufgaben sind, existieren mehr oder weniger Zellen einer Zellenart. Gewisse Arten werden erst in grösserem Ausmass produziert, wenn sie dringend gebraucht werden. Andere sind einfach als Reservezellen vorhanden.

2.1.2 Generierung und Regenerierung

Ein sehr wichtiges Konzept der Natur besteht darin, abgestorbene Zellen durch neue zu ersetzen. Stirbt eine Zelle ab, übernimmt die Aufgabe eine benachbarte Zelle. Laufend werden die abgestorbenen Zellen durch neue, herangewachsene Zellen ersetzt. Dieser Regenerierungsprozess wird vor allem bei Zellen eingesetzt, die unter hoher Belastung stehen oder für das System von grosser Bedeutung sind.

2.1.3 Das Programm der Zelle (DNA)

Wie eine Zelle arbeiten soll, steht in der so genannten DNA (deoxyribonucleic acid), dem Programm der Zelle. Mit Hilfe der DNA werden Proteine produziert, die dann die Aufgaben der Zelle übernehmen, wie Stoffwechsel, Produzieren von anderen Zellen usw. Eine Leberzelle braucht also nicht denselben DNA Abschnitt wie eine Hautzelle. Dennoch ist die ganze DNA in jeder Zelle vorhanden. Die Zelle braucht aber nur einen bestimmten Ausschnitt aus dem DNA für ihre Arbeit. Die nicht verwendeten Abschnitte der DNA sind komprimiert. Die Stammzellen (Siehe Stammzellen) sind noch nicht spezialisierte Zellen, das heisst, sie können einen grösseren Abschnitt der DNA verwenden und mehrere Arbeiten erledigen. Durch das Teilen der

Stammzellen entstehen dann immer spezialisiertere Zellen, deren DNA-Abschnitt gegenüber der Stammzelle eingeschränkter ist.

2.1.3.1 Wechseln des DNA Abschnitts

Zellen können auch den DNA Abschnitt wechseln. Somit kann sie bei Bedarf helfen, andere Aufgaben zu erledigen. Geschieht dies aber unkoordiniert, so sprechen wir etwa von Krebs oder einem Geschwür.

2.1.4 Das Immunsystem

Die Sicherheit oder Lebensfähigkeit bezüglich Krankheiten wird bei komplexeren Lebewesen durch das Immunsystem realisiert. Das geschieht durch Antikörper und den weissen Blutzellen, die eine Art Polizei bilden. Krankheiten werden durch das Immunsystem erkannt. Als Reaktion werden Antikörper produziert, welche die Erreger neutralisieren. Fremdkörper und neutralisierte Erreger werden durch die weissen Blutzellen gefressen und somit entfernt.

2.2 Ideenkatalog

Das neue Modell soll nach dem Grundsatz der Natur gebaut werden. Es existiert aus Zellen, was ein modularer Aufbau bedeutet. Die Struktur gibt auch hier dem Modell den Namen: Computing Cell Model CCM. Das CCM soll gewissermassen einem lebenden Organismus entsprechen.

Die wichtigsten Merkmale von CCM sollen hier stichwortartig aufgeführt sein:

- **Regenerierung**.
  - Das Generieren und Teilen von Zellen, das Wegräumen der abgestorben Zellen (Verwaltung der Ressourcen)
  - Erkennen von Fremdkörper und deren Bekämpfung (Security)

- **Parallele Verarbeitung** der Aufgaben (Multitasking)
  - Aufteilen, Verarbeiten und Zusammensetzen der Aufgaben
  - Autonome Anpassung an die Umgebung und an die Aufgabe (KI)

- **Lernfähigkeit**. Das System soll gewisse Abläufe selbständig optimieren, Erfahrungen sammeln und Verhaltensmuster lernen können (neuronal evolutionäres Prinzip).

2.2.1 Regenerierung

Ein sehr wichtiges Ziel ist die Stabilität und Systemsicherheit. Mit der Regenerierung von Zellen und einer geschickten Arbeitsteilung soll ein möglichst absturzsicheres Konzept realisiert werden. Mit automatischem Erkennen von Fehler und Einleiten der entsprechenden Massnahmen soll sich das System selber wieder stabilisieren können. Sterben Zellen ab (durch Fehler oder durch Angriffe von Fremdkörper), übernimmt die Arbeit eine benachbarte Zelle.

Zudem sollen Angriffe auf das System von Aussen besser erkannt werden können. Im weitesten Sinne würde dies einer Antivirus-Software entsprechen. Das System erkennt Viren automatisch und beseitigt sie durch entsprechende Gegenmassnahmen. Durch die Fähigkeit Erfahrungen sammeln zu können, soll das System gegen Virenarten immun werden.

2.2.2 Parallele Verarbeitung der Daten

Das System soll durch parallele Verarbeitung riesige Datenmengen schneller verarbeiten und analysieren können.

Die Performance des Systems soll sich nach dem Arbeitsvolumen richten. Stehen viele Aufträge an, vermehren sich gewisse Zellenarten, um der grösseren Arbeitslast entgegen zu wirken. Dagegen schrumpft das System, wenn wenig Arbeit zu verrichten ist (ähnlich wie beim Muskel). Dieses Auf- und Abbauen geschieht nicht unmittelbar, sondern ist eine träge Angelegenheit. Dieser Vorgang kann durch andere Systemkomponenten auch beeinflusst werden.

2.2.3 Lernfähigkeit

Das System soll, gegenüber heutiger Systeme, autonomer werden. Das heisst, das System soll sich den Anforderungen dynamisch anpassen können, Erfahrungen sammeln und diese Erfahrungen anwenden können. Damit soll die Lebensfähigkeit und damit auch die Lebensdauer des Systems deutlich erhöht werden. In einer ohnehin schon sehr kurzlebigen Zeit könnte so gerade in der Informatik etwas nachhaltiger entwickelt werden.

Das CCM wird durch ein neuronales Netzwerk erweitert. Dieses Netzwerk dient als Gehirn (Kommandozentrale) des Systems. Erfahrungen und verarbeitungstechnische Daten werden hier miteinander verknüpft. Das CCM wird so zu einem lernfähigen, autonomen System.

2.3 Zieldefinition

Das Cell Computing Model ist ein Konzept, welches auf dem objektorientierten Modell aufbaut. In folgenden Bereichen soll eine Vereinfachung gegenüber dem objektorientierten Modell erzielt werden:

- Schnellere Entwicklung von autonomen, lernfähigen Systemen
- Entwicklung von komplexen Systemen vereinheitlichen
- Stabile Systeme bezüglich Absturz und Angriffe von aussen
- Parallel processing für Datenverarbeitung
- Verteiltes System, welches die heutigen Techniken wie Webservices unterstützt und erweitert (XML, SOAP).
- Verwenden gängiger Netzwerke und des Internets (TCP/IP)
- Framework für künstliche Intelligenz.

2.3.1 Framework

Mit dem CCM muss ein Framework entwickelt werden, welches erlaubt, Zellen und deren DNA zu programmieren. Das Framework ist eine Software, welches dem Programmierer und Entwickler die grundsätzlichen Strukturen und Rahmenbedingungen vorgibt, und somit die Entwicklungszeit reduziert.

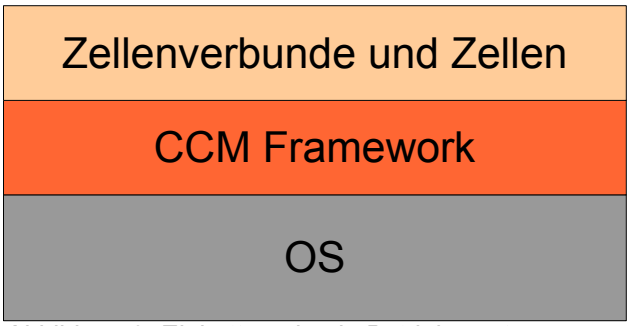

Abbildung 2: Einbettung in ein Betriebssystem

Das Framework setzt auf einem Betriebssystem auf, welches mindestens multitasking fähig und netzwerkfähig ist.

Aufgaben des Frameworks:

- Gemeinsame Basis für Kommunikation
- Verwaltung der Ressourcen
- API (Application Program Interface) für künstliche Intelligenz
- API für Zellen Programmierung (DNA)
- ev. Windows Manager (Graphische Oberfläche)

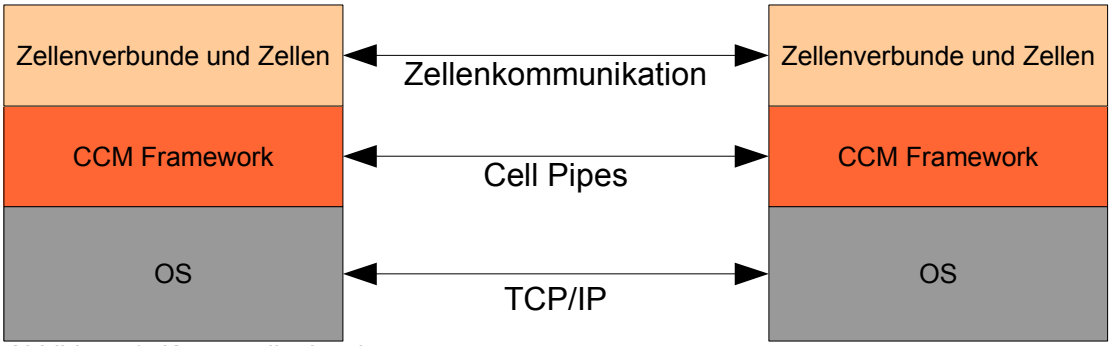

Abbildung 3: Kommunikationslayer

2.3.2 Hardware

Eine optimale Hardware für das CCM ist ein Multiprozessor-System. Jede Zelle entspricht mindestens einem Task im herkömmlichen Sinn. Will dieser Task ohne Context Switching arbeiten, also real Multitasking, muss jede Zelle

ihren eigenen Prozessor haben. Das heisst es existieren Tausende, vielleicht Millionen von Prozessoren in diesem Hardwaresystem.

Dies kann bereits mit herkömmlichen Netzwerken erreicht werden. Mehrere Computer kommunizieren über dieses Netzwerk miteinander. Das CCM wird also als ein verteiltes System betrieben.

In den Anfängen der Entwicklung des CCM kann auch mit einem Einzelprozessor gearbeitet werden. Gewisse Verhaltensweise können so bereits Umgesetzt werden. Zudem sind die Einzelprozessoren relativ kostengünstig.

2.4 Einsatzgebiet

2.4.1 Vorteile

Vorteile sind immer relativ zu den existierenden Systemen auszumachen. Ein wichtiger Punkt ist sicher die Autonomie des Systems. Es kann sich selbstständig an der Umgebung und an die Aufträge anpassen, was eine sehr hohe Flexibilität und Lebensdauer mit sich bringt. Durch künstliche Intelligenz und der Lernfähigkeit kann das System gewisse Entscheidungen selber treffen und Abläufe noch optimieren.

Die Vermehrung jener Zellen, welche sehr oft und intensiv genutzt werden, erhöht die Performance des Systems (Sofern mehrere Prozessoren zur Verfügung stehen). Somit kann das System je nach Arbeit reagieren und sich anpassen. Durch so genannte Sicherheitszellen wird das gleiche Schema für die Systemsicherheit implementiert. Je mehr das System angegriffen wird (durch Hacker, Fremdzellen usw.), desto mehr Sicherheitszellen werden produziert. Zudem kann das System gegen gewisse Fremdzellen immun werden, indem gewisse Zellenarten von Anfang an als schädlich erkannt und somit neutralisiert werden können.

Ein weiterer Vorteil liegt in der hohen Stabilität des Systems. Es können immer nur kleine Teile dieses System zerstört werden, und diese werden fortlaufend ersetzt. Es handelt sich hier um einen Regenerierungsseffekt.

2.4.2 Nachteile

Das System braucht enorme Hardware in Bezug auf Schnelligkeit und Prozessorleistung. Man kann sich etwa vorstellen, dass jede Zelle ein Thread oder sogar ein eigener Prozess im heutigen Sinne der Informatik ist. Bei herkömmlichen Systemen (PC mit einem Prozessor) wird mit context switching gearbeitet (Pseudo multi threading). Dies bringt nicht eine wesentliche Steigerung der Leistung. Im Gegenteil: bei gewissen Aufträgen verliert man Leistung durch den Wechsel von einem Thread zum anderen (Scheduling). Um eine grosse Leistungssteigerung vollbringen zu können, sollte jede Zelle einen eigenen Prozessor (Hardware) für sich beanspruchen können. Das heisst für die Hardware, dass CCM einen Parallelrechner braucht. In dieser Hinsicht ist der Erfolg von CCM von zukünftigen Hardwareentwicklungen abhängig (Siehe Zukunftsvisionen).

Bei sequenziellen Systemen kann zu jedem Zeitpunkt genau gesagt werden, welches der nächste Verarbeitungsschritt ist. Auch bei der OOP ist genau definiert, welches Objekt mit welchem Objekt wie kommuniziert. Im CCM ist dies nicht mehr eindeutig feststellbar. Welche Zelle nun gerade welchen Auftrag verarbeitet, kann nicht vorausgesagt werden. Auch wie viel das System wächst bzw. schrumpft kann höchstens geschätzt werden. Es kann höchstens ein Trend festgestellt werden. Wie beim Wetter, bei welchem man auch nie hundertprozentig vorhersagen kann, wie der morgige Tag aussieht. Dieses Verhalten könnte als ein Nachteil empfunden werden.

2.5 Zukunftsvisionen

Wie sich die Computertechnologie in mittlerer Zukunft entwickelt, lässt sich nicht so leicht analysieren. Sicher ist, dass mit der neuen Quanten- und Nanotechnologie ein gewaltiger Schritt in Richtung Kleinstprozessoren gemacht wird. Eine meiner Visionen ist, dass sich in Zukunft ein PC nicht nur aus einem Einzelprozessor aufbaut, sondern dass man, ähnlich wie beim RAM-Speicher, Processor Banks kaufen kann. Auf solchen Banks befinden sich unzählige Prozessoren, die von einem Programm (oder eine Zelle) aktiviert werden, und nun die Aufgabe dieses Programms bewältigen. Ähnlich wie ein Speicherbaustein kann nun eine Zelle einen freien Prozessor aktivieren. Werden Prozessoren nicht gebraucht, befinden sie sich im Leerlauf. Diese Technologie würde es dann erlauben, eine Menge von Prozessen in Echtheit parallel arbeiten zu lassen. Das CCM würde sich hervorragend für solche Hardware eignen, da es für das parallele Abarbeiten von Aufträgen konzipiert ist.

Zukünftig sollen alle Computersysteme mit CCM ausgestattet werden. Zusammen mit der Intelligenz des Erfahrungssystems (Gedächtnis des Systems) könnte das System die neue Computersoftwarerevolution bedeuten. Computer nehmen dem Menschen nicht nur rechnerische und datenverwaltende Aufgaben ab, sondern nun auch das Treffen von minderwertigen Entscheidungen. Sie werden auch hier nicht, wie schon vor etlichen Jahren befürchtet, den Menschen ersetzen, sondern ihm lästige oder zeitkritische Arbeit und Entscheidungen abnehmen. Lernfähige Systeme werden in der Wirtschaft wie auch für militärische Zwecke gebraucht, um aus Erfahrungen und neuen Gegebenheiten Entscheidungen treffen zu können.

3 Grundkonzept des CCM

Der folgende Abschnitt zeigt eine mögliche Umsetzung des Cell Computing Model auf. Dabei wird immer wieder auf die Synergien zur realen Welt zurückgegriffen. Gerade dieser Abschnitt kann sich im Verlaufe der Entwicklung noch stark verändern, da viele Strategien zuerst noch getestet und erforscht werden müssen. Es ist auch nicht ausgeschlossen, dass neue Erkenntnisse aus der Genforschung die Entwicklung dieses Projekts beeinflussen werden. Deshalb kann es sich hier nicht um eine endgültige Lösung handeln. Vielmehr soll es zum Fördern von Ideen und neuen Ansätzen dienen.

3.1 Das EVA-Prinzip

Die Grundidee des Informatiksystems bleibt auch hier bestehen. Es handelt sich also ebenfalls um ein EVA-System (Eingabe-Verarbeitung-Ausgabe). Das CCM Konzept beinhaltet eine Dateneingabe, Datenverarbeitung und eine Datenausgabe. Der grosse Unterschied ist aber in der Systemarchitektur und der Art der Datenverarbeitung zu finden. Im Weiteren wird die Peripherie anders benannt. So heissen die Eingabegeräte mit dem entsprechenden Zellenhaufen **Sensoren** (Kamera, Maus, Tastatur etc.), die Ausgabegeräte werden als **Aktoren** bezeichnet (Drucker, Bildschirm, Roboterarm etc.). Diese Bezeichnungen sollen abstrakt verstanden werden. Es können sowohl virtuelle wie auch reale Geräte sein. Durch die Sensoren kann das System zu Daten gelangen, während bei Aktoren irgend ein Resultat der Eingabedaten und deren Verarbeitung sichtbar wird, zum Bsp. Bewegung eines Roboters.

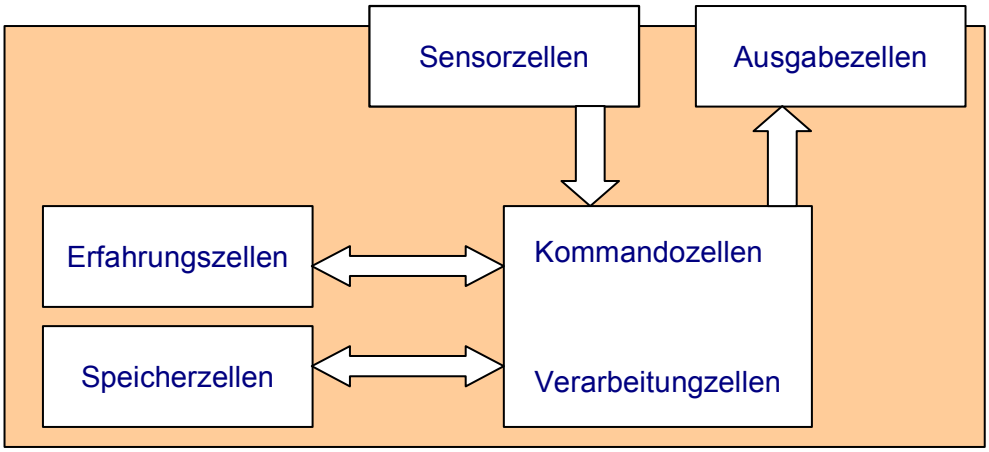

Abbildung 4: Das EVA Prinzip

Das Prinzip wird mit Hilfe von Tastaturen und Bildschirmen verdeutlicht:

An einem solchen Grosssystem können zum Beispiel hunderte Tastaturen vorhanden sein. Jede spricht eine Sensorzelle an. Diese Zellen nehmen die Eingaben entgegen und speisen damit das System. Jeder Bildschirm wird von einer oder mehreren Ausgabezellen angesprochen. Das System kann nun Informationen und Daten über diese Ausgabezellen an den Bildschirm weitergeben. Dabei könnte es sich auch um hunderte von Bildschirmen handeln.

Wo diese Zelle lebt kann je nach Hardware verschieden sein. Sie könnte zum Beispiel direkt in der Tastatur existieren. Das hätte den Vorteil, dass der Treiber sich bereits bei der richtigen Hardware befindet. Kurz gesagt: die Software und Hardware verschmelzen zu einer Einheit.

Die Steuerung des Systems erfolgt durch eine oder mehrere Kommandozentrale(n). Die erlernten Erfahrungswerten werden hier verknüpft und verwaltet. Durch Kommunikationssysteme kann die Zentrale die einzelnen Zellenhaufen steuern und so Einfluss auf den Verarbeitungsprozess nehmen. Damit kann die Möglichkeit einer autonomen Systemverbesserung und Optimierung erzielt werden, was wiederum zu einer besseren Lebensfähigkeit führt. Ein weiterer Ansatz zur Prozessoptimierung ist das Züchten von guten und das Zerstören der weniger effektiven Zellen (evolutionäres Prinzip).

Wie sieht es bei der Datenablage aus? Die Datenablage ist einem Datenarchiv ähnlich. Durch Verzeichnisse kann der Benutzer Strukturen anlegen und darin seine Dokumente speichern. Hier wird das CCM keine Änderung bringen. Im Gegenteil sollen hier die vorhandenen Technologien weiter verwendet werden.

Irgendwo müssen die Zellen ihre Arbeiten holen bzw. speichern, Daten im RAM zwischenspeichern und Erfahrungen verknüpfen. Aber auch hier können wir vorhandene Technologien verwenden, die sich gerade in der objektorientierten Programmierung bewährt haben. Im CCM unterscheiden wir folgende Datenarten:

- **Zellendaten**. Diese Daten verwendet eine Zellen nur intern. Sie sind die Daten der Zellenattribute. Diese Daten sind von aussen nicht zugänglich.

- **Speicherdaten**. Diese Daten entsprechen sozusagen den gespeicherten Dokumenten und der verarbeitete Daten des Benutzers.

- **Erfahrungsdaten** sind Einstellungen und Erfahrungen des Systems. Diese Daten werden von der Kommandozentrale verwaltet. Diese Daten könnten zum Beispiel in einem neuronalen Netz verknüpft gespeichert werden.

4 Das Zellenleben

Dieser Abschnitt stellt zwei Möglichkeiten vor, Zellen zu generieren. Bei der ersten Variante werden die Zellen durch eine Zellenfabrik erzeugt. Bei der zweiten Variante wird das Klonen angewandt. Beide Varianten werden auch von der Natur eingesetzt, um Zellen zu produzieren. Dabei ist entscheidend, wo und wie die Zellen eingesetzt werden, und damit natürlich um welche Zellenart es sich handelt.

4.1 Der Lebenszyklus

Jede Zelle durchlebt einen Zyklus. Ein Zyklus umfasst folgende Stadien:

- Zelle wird generiert
- Zelle ist inaktiv (dient als Reserve)
- Zelle wird aktiviert und kann nun Daten verarbeiten
- Zelle wird wieder inaktiv wenn keine Daten vorhanden sind
- Zelle stirbt ab, wenn sie
  - Fehlerhaft ist
  - Gefährlich ist
  - Zu lange inaktiv ist (verhungert)

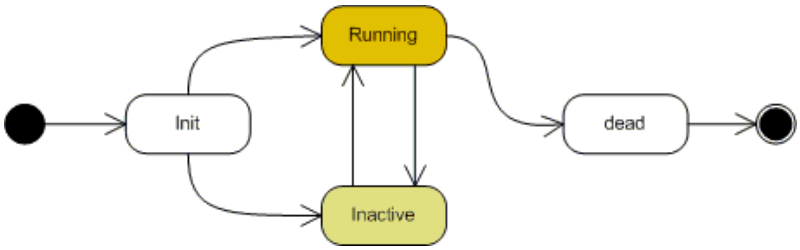

Abbildung 5: Zustände einer Zelle

In einem multitaskfähigen Betriebssystem können Zellen als Prozesse oder Threads implementiert werden.

4.2 Erzeugen und Abbauen von Zellen

Das System soll sich dynamisch bezüglich der Eingangslast verhalten. Dazu muss ein Verfahren gefunden werden, welches sich möglichst stabil verhält.

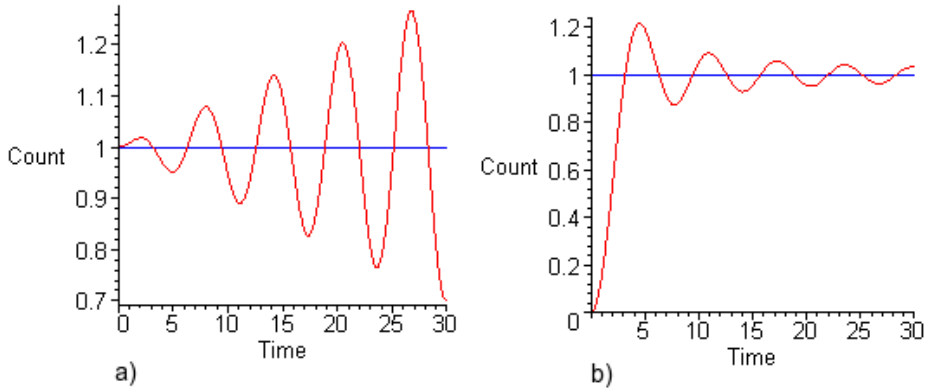

Abbildung 6: Zeigt die Verhaltensweisen von Klonalgorithmen welche mit der Zeit instabil werden (a) oder sich an ein Optimum angleichen (b).

4.2.1 Erstellen von Zellen / die Zellenfabriken

Die neuen Zellen werden von einer Zellenfabrik erzeugt. Diese Zellenfabrik ist ein Zellenhaufen, welcher speziell zum Bau von anderen Zellen konstruiert wurde. Wann die neuen Zellen erstellt werden, kann nicht eindeutig definiert werden. Es gibt Zellen, die eine Herstellungspriorität haben, aber schlussendlich entscheidet die Fabrik, welchen Auftrag sie zuerst erledigt. Sobald die Zelle erstellt wurde, wird sie automatisch in Betrieb genommen, und kann nun ihre Aufgabe aufnehmen.

Die Zellenfabriken werden durch ein weiter unten beschriebenes Botschaftssystem benachrichtigt, welche Zellenart erstellt werden muss. Sind gewisse Zellen voll ausgelastet, werden Botschaften ausgeschüttet, die von den entsprechenden Fabriken verstanden werden. Die Fabrik reagiert darauf und generiert neue Zellen, welche die überlasteten Zellen unterstützen sollen. So kann das System je nach Nachfrage und Auslastung selber reagieren. Aus Sicherheitsgründen hat jede Nachfrage eine obere Grenze, die das System nicht überschreiten darf. Folgendes Beispiel soll dies verdeutlichen:

Eine Zelle ist falsch programmiert und deshalb immer voll ausgelastet. Die Fabrik erzeugt nun weitere Zellen, die die Arbeit der überlasteten Zelle versuchen zu teilen. Jede Zelle ist nun durch diesen Fehler auch voll ausgelastet. Solche Zellenarten nennen wir Wucherzellen. Ohne obere Schranke würde die Fabrik immer mehr Zellen produzieren, bis die Ressourcen des Systems aufgebraucht worden sind. Solche Zellen werden vom System speziell behandelt, wenn das System diese als Wucherzelle erkennt.

Es gibt eine weitere Art von Zellen, die egoistischen Zellen. Dies sind Zellen die Ihre Arbeit nicht teilen können oder wollen. Sie sind weit weniger schlimm als Wucherzellen. Von ihrer Gattung existieren noch inaktive Zellen, und somit generiert die Fabrik keine neuen Zellen. Einzig die Performance der Arbeit leidet darunter. Die Philosophie des System wird hier nicht eingehalten, die Arbeit zu teilen und somit schneller zum Ziel zu kommen (Siehe Arbeitsweise von Zellen).

Das Prinzip der Zellenfabriken kann vor allem für dynamische Zellen verwendet werden. Die dynamischen Zellen sind Zellen, die nicht fest zu einem Zellenhaufen gehören, sondern ähnlich wie Blutzellen zum Beispiel, im System Botschaften und Daten übertragen. Oder Zellenarten wie die Fresszellen und Polizeizellen, welche durch eine Fabrik in das System eingeschleust werden.

4.2.2 Klonen von Zellen

Zellen haben die Fähigkeit sich selber auch zu vermehren. Es gibt so genannte Mutterzellen, die sich bei Überlastung teilen können. Pro Zellensystem existiert eine Mutterzelle, die die gesamte Fähigkeit des Teilsystems besitzt. Durch das Teilen entstehen neue Zellen, die beim Erledigen der Aufgabe mithelfen. Die geklonten Zellen sind spezialisierte Zellentypen, die nur ein ganz bestimmten Teil einer Aufgabe erledigen können. Ihre Fähigkeit ist gegenüber der Mutterzelle eingeschränkt. Sie können aber bei Bedarf auch ihren spezialisierten Teilbereich wechseln, und so anderen Zellen in diesem Zellensystem aushelfen.

Das Prinzip des Klonen kann vor allem bei statischen Strukturen (Zellenhaufen) eingesetzt werden. Stellt man sich ein Baum mit verschiedenen Knoten vor, so kann der Knoten sich nun teilen und vermehren. Hier wäre eine Zellenfabrik weniger geeignet, da die Fabrik wissen müsste, zu welchem Zellenhaufen die Zellen zu erstellen sind, und wie sie dort angebunden werden müssten. Die Mutterzelle hingegen ist bereits im Haufen vorhanden und kennt die Struktur, bzw. den Ort, wo die neue Zelle ihren Platz findet.

4.2.2.1 Vermehrung in dem Zellenhaufen

Gehen wir einmal davon aus, dass die Arbeit, welche eine Zelle verrichten muss, in Form von Meldungen (Botschaften) existiert. Wir benötigen dazu einen Eingangspuffer, in welchem alle Meldungen zwischengespeichert werden, welche noch nicht von der Zelle verarbeitet wurden. Die Grösse des Puffers dient als wichtigste Grösse, um die Auslastung zu berechnen.

Der Algorithmus benötigt folgende Eingabeparameter

- Anzahl Meldungen eines Zellensystems zu einem beliebigen Zeitpunkt t: $d(t)$
- Maximale Grenze des Meldungseingangs (Puffers) N_{MAX}
- Reaktionsgrenze N_{react}

Dabei wird der Zuwachs der Meldungen und die Änderung des Zuwachs geprüft, wenn die Grösse des Eingangspuffer die Reaktionsgrenze überschreitet. Zellen werden geklont, wenn der Zuwachs positiv und die Änderung des Zuwachs stabil oder zunehmend ist.

Die Funktion $f_k(n)$ bestimmt, wann eine Zelle im Zellensystem geklont werden soll. Resultiert eine 1 wird eine neue Zelle generiert, bei 0 wird das System nicht verändert.

$$f_k(t) = \begin{cases} 1 & , \textbf{\textit{if}}\ (\ddot{d}(t) \geqslant 0 \wedge \dot{d}(t) > 0 \wedge d(t) > N_{react}) \vee d(t) > N_{MAX} \\ 0 & , \textbf{\textit{else}} \end{cases}$$

In der obigen Formel ist die Messzeit analog. Die Messzeit muss für den Computer noch diskretisiert werden. Halten wir die Messzeit variabel, so passt sie sich der Reaktionszeit des Zellenhaufens an. Für die Messzeit können eine obere und untere Schranke angegeben werden, T_{MAX} und T_{MIN}. Dies ist notwendig, um auf variable Verarbeitungszeit der Meldungen reagieren zu können. Die Messzeiten haben einen direkten Einfluss auf die Reaktionszeit des Systems. Bei langer Verarbeitungszeit der Meldungen macht es wenig Sinn, schnelle Messungen durchzuführen. Dies würde höchstens die Performance des Systems beeinträchtigen. Im Gegensatz ist es nicht optimal, bei kurzer Verarbeitungszeit der Meldungen nur gelegentlich zu Messen. Das Wachstum würde sehr träge sein und es würde lange dauern, bis das System sich stabilisiert. Deshalb ist eine variable Messzeit sehr nützlich, um das System zeitoptimal zu steuern.

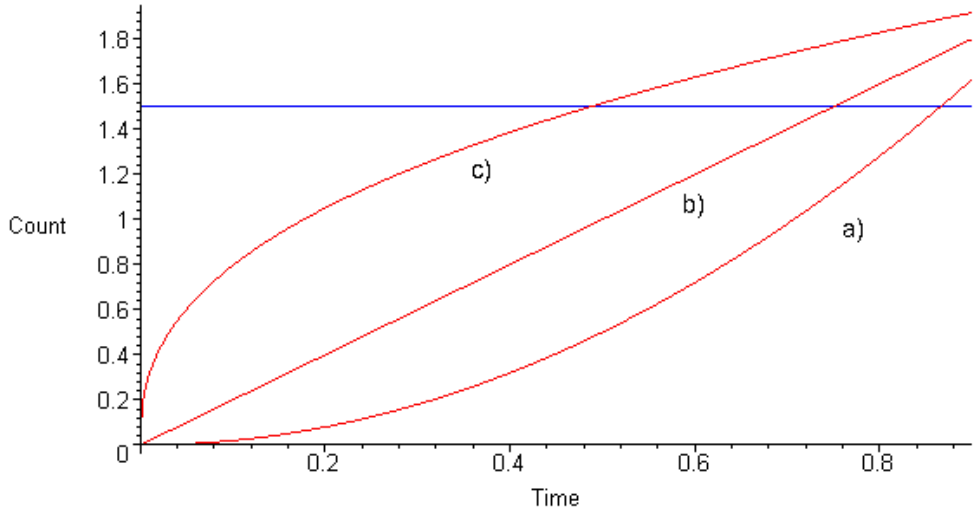

Abbildung 7: Funktionsweise des Klonalgorithmus. Ist ein Anstieg der Meldungen vorhanden, wird geprüft, ob die Veränderung positiv (a) oder gleich bleibend ist (b). Nur in diesen 2 Fällen wird geklont. Im Falle (c) wird erst geklont, wenn die Grenze überschritten wurde.

Gibt die Funktion $f_k(n)$ den Wert 1 zurück, was bedeutet, dass die Zelle geklont wird, so kann nun eine noch inaktive Zelle wieder aktiviert werden. Sind keine inaktiven Zellen mehr vorhanden, dann muss eine neue gleiche Zelle erstellt werden.

4.2.3 Abbauen von Zellen

Die Lebensdauer einer Zelle ist nicht eindeutig festlegbar. Werden Zellen über längere Zeit nicht gebraucht, werden sie zuerst inaktiv, später sterben sie ab. Dadurch kann das System auch langfristig Ressourcen freigeben, wenn die Auslastung tief ist.

Dieser Prozess schliesst den Kreislauf der Zellen. Das System wächst also, wenn die Auslastung gross ist und schrumpft, wenn die Auslastung tief ist. Diese Veränderung darf natürlich nicht schnell erfolgen, sondern entspricht einen Trend über längere Zeit.

4.2.3.1 Die Lebenskraft

Jeder Zelle wird eine Lebenskraft zugewiesen, welche beim Erstellen der Zelle das Maximum L_{MAX} enthält (zum Beispiel den Wert 100). Die Lebenskraft wird durch Verarbeiten von Meldungen um einen konstanten Wert erhöht L_{add} .

Ist keine Meldung zum Bearbeiten vorhanden, hungert die Zelle, sie liegt brach. Dabei wird die Lebenskraft um einen konstanten Wert verringert L_{sub} . Mit abnehmender Lebenskraft nimmt die Wahrscheinlichkeit zu, dass die Zelle inaktiv wird.

Die Funktion $h(t)$ liefert die Anzahl Arbeitspakete, die von einer Zelle bearbeitet wurden.

Der Algorithmus verwendet folgende Eingabeparameter:

- Lebenskraft der Zelle in Abhängigkeit der Zeit t: $g(t)$
- Erhöhung der Lebenskraft L_{add}
- Verringerung Lebenskraft L_{sub}
- Faktor für Wahrscheinlichkeit des Abbaus ω

$$g(t) = g(t-1) + \begin{cases} L_{add}, if\ h(t) > 0 \\ -L_{sub}, else \end{cases}$$

Die Funktion $f_a(t)$ gibt 1 zurück, wenn die Zelle inaktiv werden muss, sonst 0.

$$f_a(t) = \begin{cases} 1, if\ L_{MAX} \cdot e^{-g(t)/\omega} > \psi(L_{MAX}) \\ 0, else \end{cases}$$

wobei Ψ eine Random-Funktion ist die einen Wert zwischen 0 und L_{MAX} zurück liefert.

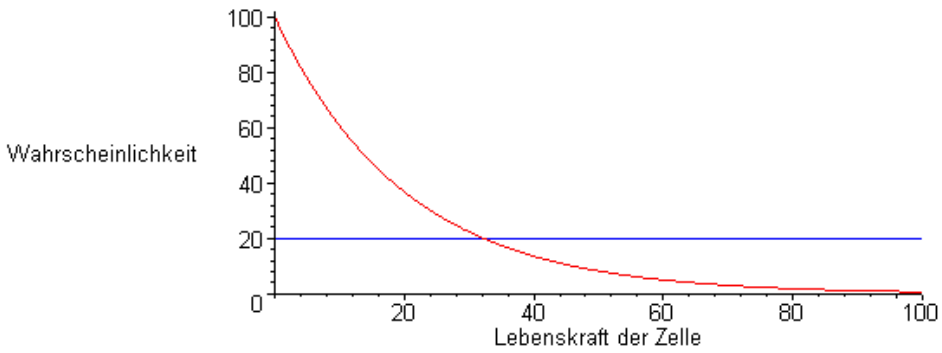

Abbildung 8: Darstellung der Wahrscheinlichkeitsfunktion in Bezug zur Lebenskraft einer Zelle

Durch eine simple Auszeit (Expire Time) werden die inaktiven Zellen dann schlussendlich zerstört.

4.2.3.2 Absturz durch Fehler

Zellen können aber auch sterben, wenn sie einen groben Fehler im System verursachen, oder wenn sie wegen eines Programmierfehlers abstürzen. Das heisst, was heute in der Computerwelt zu einem Programmabsturz führt, führt hier zum Absterben einer Zelle.

Einen anderen schwerwiegenden Fehler kann geschehen, wenn eine Zelle korrupte Daten produziert, oder durch einen Angriff von Aussen infiziert wurde. Sie überschreitet dadurch ihre Kompetenz und wird als Fremdkörper im System erkannt. Ebenso werden Zellen unbekannter Herkunft als Fremdkörper identifiziert. In diesem Falle kommen die Polizeizellen zum Einsatz. Diese Zellen sind ausschliesslich damit beschäftigt, die Sicherheit im System zu gewährleisten. Alle Zellen werden durch die Polizeizellen analysiert. Halten sie bestimmte Richtlinien nicht ein oder überschreiten ihre Kompetenzen, werden sie neutralisiert. Viren und Krankheiten werden als Erfahrungswerte gespeichert, damit sie beim Wiederauftreten schneller erkannt werden können. Die neutralisierten Zellen gelten dann ebenfalls als abgestorbene Zellen. Das Sicherheitssystem soll wie ein Immunsystem funktionieren und ist im weitesten Sinne ein integriertes Antivirus-Programm.

Die abgestorbenen Zellen werden von Fresszellen gefressen, und die Ressourcen werden dem System zurückgegeben. Wie jede Zellenart werden auch die Fresszellen je nach Bedarf von einer Fabrik erzeugt. Sterben Fresszellen ab, werden sie von ihren Artgenossen ebenfalls weggeräumt. Die Fresszellen können auch als Destruktoren bezeichnet werden. Sie bilden den Gegenpol zu den Fabriken und dem Klonen und halten so das Gleichgewicht.

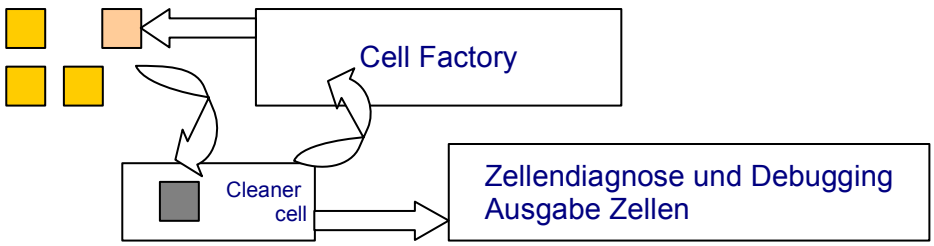

Abbildung 9: Schematische Darstellung der Funktionsweise einer Zellenfabrik

Durch ein evolutionäres Prinzip kann das Gesamtsystem gewisse Zellen verbessern, verändern oder sogar züchten. Dabei sollen unter anderem Daten beim Absterben einer Zelle generiert werden. Diese Daten stehen nach aussen zur Verfügung und können von Fresszellen gelesen und analysiert werden. Diese Daten werden dann zum Verbessern aber auch zum

Debuggen des Systems verwendet.

4.3 Der Regenerierungseffekt

Der Begriff Regenerierung steht für die Fähigkeit eines Systems oder Organismus, sich selber zu reparieren, wenn irgendwo durch äussere Einwirkungen ein Schaden entstanden ist. Dabei ist es wichtig, dass der Schaden sich möglichst nur lokal auswirkt, und das System sofort wieder einen stabilen Zustand erreicht. Das Prinzip der Zellen ist hier sehr geeignet, da die Abstützung sehr breit ist. Ähnlich wie beim Internet: steigt irgendwo ein Server aus, stirbt nicht gerade das ganze Netz.

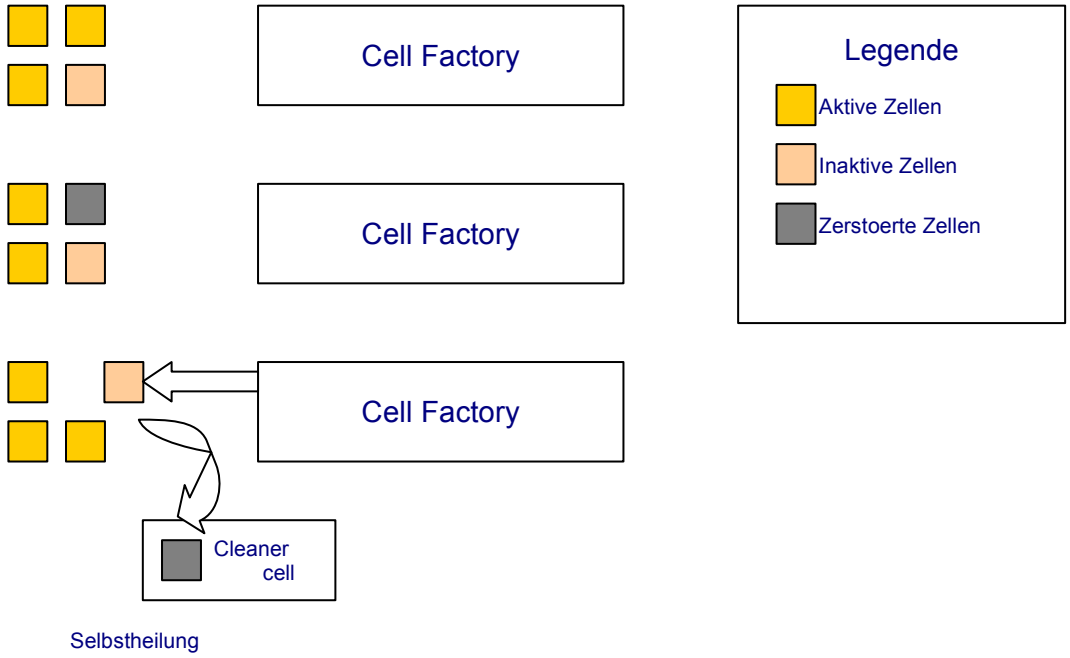

Abbildung 10: Regenerierung von Zellen. Mit Hilfe von Zellenfabriken und Fresszellen wird der Lebenszyklus von Zellen beeinflusst. Nach dem gleichen Prinzip geschieht die Regenerierung durch Klonen.

In dem CCM sieht der Regenerierungsprozess folgendermassen aus: steigt eine Zelle aus, muss diese Zelle ersetzt werden, damit das System weiterarbeiten kann. Um das Problem kurzfristig in den Griff zu bekommen und keinen grossen Leistungsverlust erleiden zu müssen, übernimmt eine benachbarte Zelle die Arbeit der defekten (zerstörten) Zelle. Das bedingt, dass es im System noch inaktive Zellen hat, welche bei Bedarf sofort eingesetzt werden können. Gerade bei wichtigen Aufgaben ist es wichtig, dass die verstorbenen Zellen sofort ersetzt werden können, und nicht auf die

Erzeugung neuer Zellen gewartet werden muss.

5 DNA-Der Programmcode der Zelle

Wie weiss nun die Zelle, wie sie eine Aufgabe wahrnehmen und erledigen soll? Jede Zelle besitzt ein DNA (*deoxyribonucleic acid*). Die DNA entspricht dem Programm der Zellen. Dieses Programm beschreibt die Verhaltens und Verarbeitungsalgorithmen. Jede Zelle besitzt immer die ganze DNA des Systems. Mit anderen Worten könnte eine Zelle jede Aufgabe wahrnehmen. Durch den Teilungsprozess werden jedoch den einzelnen Zellen Bereiche der DNA zugeordnet. Die Zellen kennen zwar die ganze DNA, arbeiten aber ausschliesslich im zugeordneten Bereich. Theoretisch wäre es denkbar diesen Bereich beliebig zu wechseln. Um die DNA zu interpretieren existiert in jeder Zelle ein DNA-Interpreter. Dieser Interpreter erstellt nach der DNA die Algorithmen, welche die Zelle verwendet, um ihre Aufgaben wahrzunehmen. Die DNA beschreibt auch wie diese Algorithmen mit einander agieren und kommunizieren.

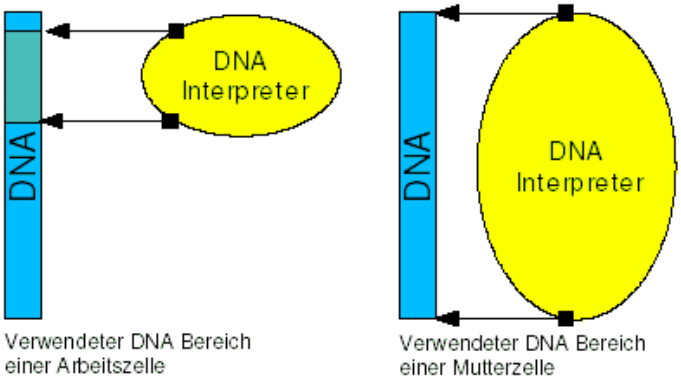

Abbildung 11: Funktionsweise eines DNA-Interpreter. Bei spezialisierten Zellen (links) wird nicht die gesamte DNA verwendet. Mutterzellen können Aufgaben von verschiedenen Zellen übernehmen (rechts), da sie grössere Ausschnitte aus der DNA interpretieren können.

Wie die DNA im Detail realisiert wird, ist noch offen. Man könnte die gesamte Information jeder Zelle mitgeben. Dieses Prinzip wird in der Natur verwendet. Der Nachteil ist, dass dies enorm viel Speicherplatz verschwendet. Die Natur komprimiert deshalb den nicht verwendeten Teil der DNA. Eine weitere Möglichkeit wäre das Halten von globalen Daten. Dadurch würde das Speicherproblem weitgehend gelöst. Es hat aber der grosse Nachteil, dass sich die DNA durch die zentrale Haltung ein gutes Angriffsziel ist. Zudem ist hier die Realisierung von Verteilten Systemen sehr aufwendig.

6 Immunsystem-Sicherheit im System

Ein Immunsystem dient dem automatischen Erkennen von Viren (schädlichen Programmen) und trojanischen Pferden (Spionage). Schlussendlich muss das Immunsystem die geeigneten Gegenmassnahmen treffen. In diesem Kapitel werden einige Verfahren vorgestellt, wie ein solches Immunsystem im Cell Computing Model realisiert werden könnte.

6.1 Erkennen von Fremdzellen

Fremdzellen sind Zellen, die nicht zum eigentlichen System gehören und somit als schädlich oder unsicher eingestuft werden. In der Biologie werden solche Zellen anhand der molekularen Struktur erkannt, sie riechen anders als die körpereigenen Zellen. Dies ist in der Informatik nicht so leicht nachzubilden. Wir können aber eine verwandtes Prinzip anwenden, den Fingerabdruck.

Mit einem Fingerabdruck (einer One-Way-Hash-Function) können wir den Programmcode einer Zelle eindeutig erkennen.

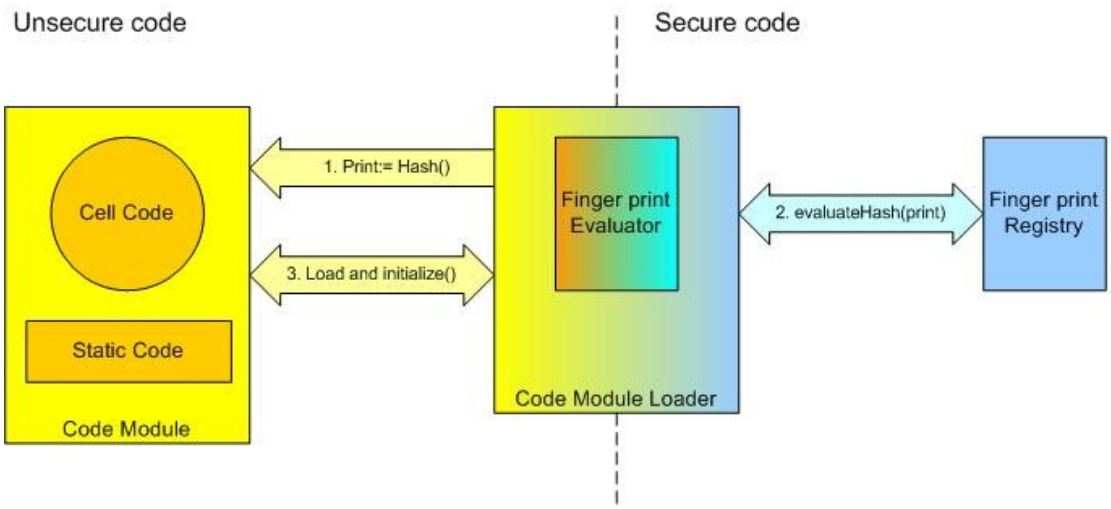

Abbildung 12: Nach dem Berechnen und Vergleichen des Finger Print kann das Code-Modul mit der Zelle geladen werden.

Dabei prüfen wir nicht nur den Programmcode der Zelle, sondern auch den zusätzlichen statischen Programmcode, da sich auch hier schädliche Programmteile verstecken könnten (Abbildung 12). Beide, der Programmcode der Zelle und der statische Programmcode, sind in einem

Code-Modul zusammengefasst. In diesem Zusammenhang sprechen wir in der Informatik auch von Packages oder Shared Libraries.

Der Fingerabdruck (Finger print) wird nach dem Berechnen mit einem Eintrag einer Datenbank (Finger print registry) verglichen. Stimmt der Fingerabdruck mit dem registrierten Fingerabdruck überein, kann das Code-Modul geladen werden. Stimmt der Fingerabdruck nicht mit dem Eintrag der Datenbank überein, wird das gesamte Code-Modul als unsicher eingestuft. Das Code-Modul kann aber trotzdem geladen werden, wenn der Entwickler des Systems dies zulassen möchte.

6.2 Mobile Sicherheitszellen

Das Erkennen der Fremdkörper ist in dem Zellensystem ein wichtiger Bestandteil, um das System längerfristig am Leben zu erhalten. Um die Sicherheit möglichst breit abzustützen, wird eine dezentrale Eingreiftruppe geschaffen. Sie ermöglicht es, schnell und effizient einzugreifen, wenn ein Angriff auf das System erfolgt. Diese Eingreiftruppe besteht aus so genannten Sicherheitszellen, die andere Zellen überwachen und analysieren können. Diese Zellen können sich im System frei herum bewegen, sie sind mobil.

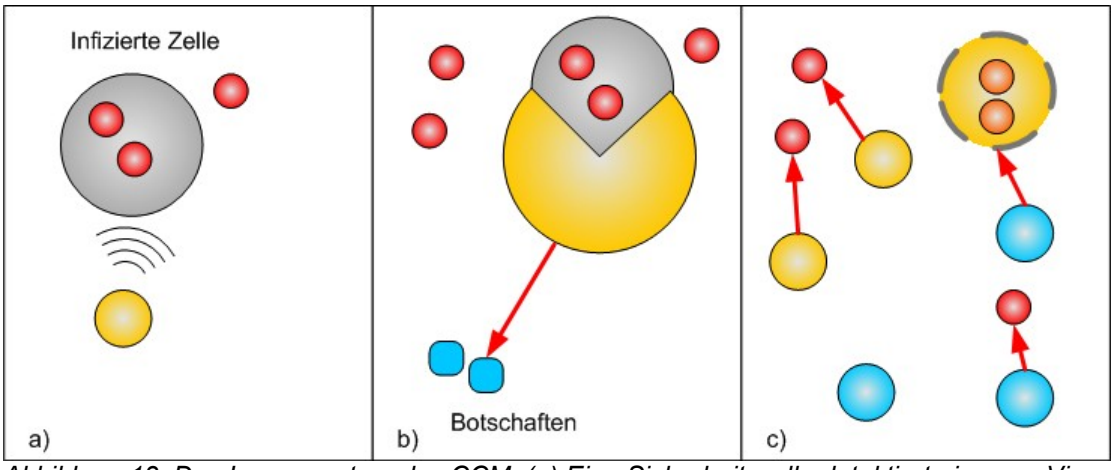

Abbildung 13: Das Immunsystem des CCM. (a) Eine Sicherheitszelle detektiert eine von Viren befallene Zelle. (b) Die Sicherheitszelle sendet Botschaften, welche andere Sicherheitszellen anlocken und versucht die befallene Zelle zu vernichten. (c) Spezialisierte Sicherheitszellen neutralisieren die Viren und die defekten Sicherheitszellen.

Beim Auftreten eines Problems können durch Trendkommunikation Sicherheitszellen angelockt werden, die versuchen das Problem zu analysieren und erste Massnahmen zu ergreifen. Eine solche Analyse kann

unter anderem auch die Prüfung des Fingerabdrucks (siehe oben) beinhalten. Kann das Problem nicht ausreichend schnell oder effizient gelöst werden, können spezialisierte Sicherheitszellen angefordert werden. Das geschieht wieder durch die Trendkommunikation zu nahe liegenden Sicherheitsstellen (Zellenfabriken die Sicherheitszellen produzieren).

So genannte Gedächtniszellen, welche den Vorfall miterleben und überleben, speichern die Signatur der Virenzellen ab. Durch ihre Alarmbereitschaft, können nun Virenzellen schneller detektiert und bekämpft werden.

7 Kommunikation zwischen den Zellen

Ein sehr wichtiges Konzept ist die Kommunikation. Einerseits ist es wichtig, Zellen mit Informationen und Daten zu versorgen, die sie verarbeiten sollen (die Nahrung der Zelle), und anderseits müssen die Zelle dem System Rückmeldungen geben können. All dies geschieht mittels Botschaften (Events). Die Botschaften sind einfache Zellen oder Objekte, die Daten enthalten.

Es gibt zwei Arten von Kommunikationssysteme:

- Das Direct Communication System (DCS)
- Das Open Communication System (OCS)

7.1 Direct Communication System DCS

Das Direct Communication System DCS dient der gezielten und schnellen Kommunikation zwischen einzelnen Zellensystemen. Um ein bestimmtes Zellensystem zu erreichen muss eine Adressierung erfolgen. Der Kommunikationspartner muss bekannt sein.

Das DCS eignet sich besonders zum Steuern der Zelle oder zum Senden von Botschaften nach einem Prozessplan (Workflow).

7.1.1 Steuern von Zellen durch eine Zentrale

Die Zellensysteme werden von der Zentrale aus geleitet. Diese Zentrale hat direkten Zugang zu den Erfahrungswerten. Je nach Erfahrung und Grundintelligenz (Grundeinstellungen) des Systems, kann es nun reagieren und die Verarbeitungszellen steuern (Siehe Kommandozentrale). Rückmeldungen von Zellenfabriken und Verarbeitungszellen, sowie Daten von Sensoren können somit in die Kommandozentrale gelangen.

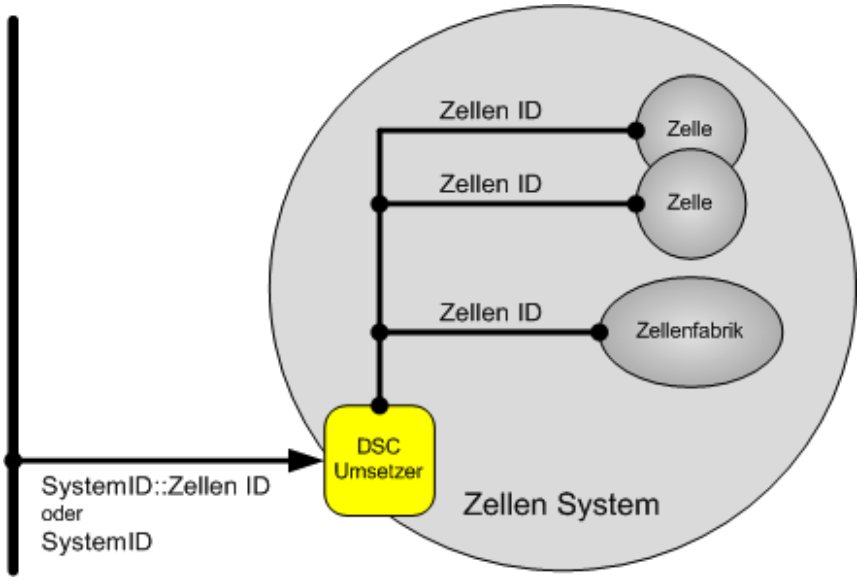

Abbildung 14: Das Direct Communication System DCS dient als direkter Draht zu Zellensystemen. Es dient der schnellen Kommunikation zwischen Zellen und Kontrollzentren

Design Hinweis: Für eine mögliche Umsetzung könnte das Soap-Protokoll von Nutzen sein. Die Adressierung der einzelnen Systeme und Zellen kann dem TCP-IP Standard angelehnt sein.

7.1.2 Workflow

Workflows finden wir im täglichen Leben. In einer Firma sind die Workflows beliebte Instrumente, um den Werdegang eines Produkts visuell darzustellen. Hier beschreibt der Workflow, wann welches Teilprodukt wo verarbeitet werden soll. Es gibt Stellen, wo eine Abteilung auf eine andere warten muss, um bestimmte Teilprodukte wieder zu eine Ganzen zusammen zu bauen. Auch können redundante Wege definiert werden, Stellvertreter die im Falle von Krankheit oder Ferien einspringen können.

Im Cell Computing Model können wir auch Workflows definieren, um dem System zu sagen, wann welches Arbeitspaket wohin gesendet werden soll. Der Workflow wird also mit dem Arbeitspaket mitgeliefert. Nähere Informationen sind im Kapitel „Arbeitsweise der Zellen“ zu finden.

7.2 Open Communication System OCS

Das Open Communication System dient im Gegensatz zum DCS als Trendkommunikation (langsame Kommunikation). Die Zellen werfen Meldungen in das Central Communication Network aus. Diese Meldungen sind nicht explizit adressiert. Jeder kann die Meldungen lesen. Wer sie interpretieren und umsetzen kann, entfernt die Meldung (verspeisen).

Ein Vorteil des OCS ist, dass verschiedene Zellen oder Zellensysteme auf Meldungen fast gleichzeitig reagieren können. Solche Meldungen werden auch fortlaufend von den Fresszellen abgebaut, wenn sie nicht gelesen werden.

Dieses System hat aber auch einen Nachteil. Der Sender hat keinen Einfluss, wann und wo seine Meldungen gelesen werden. Es muss mehrmals dieselbe Meldung versendet werden, da unter Umständen nie alle Meldungen ihr Ziel erreichen. Ob eine Meldung verstanden und umgesetzt wurde, kann der Sender erst feststellen, wenn im System die gewünschte Reaktion erfolgt.

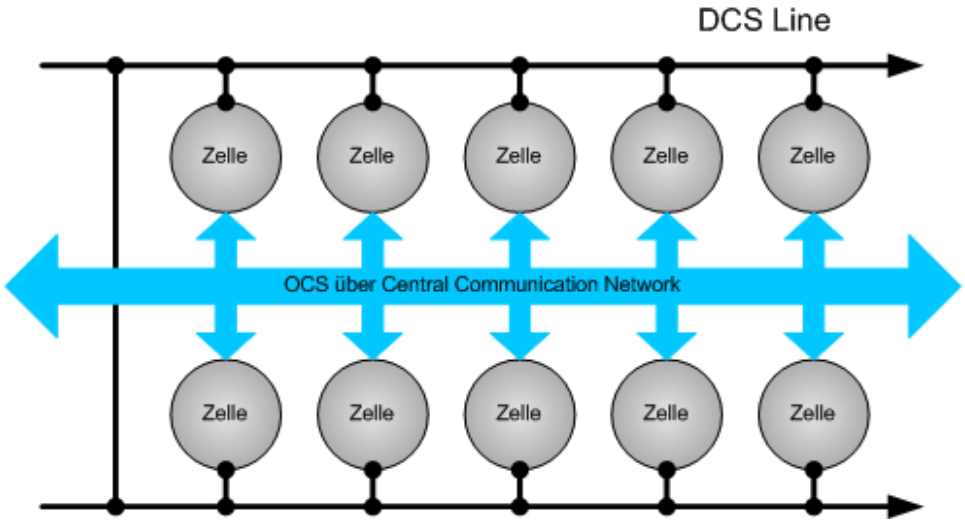

Abbildung 15: Das Open Communication System OCS dient zur Trend Kommunikation, welche längerfristige Änderungen im System steuert.

7.2.1 Das OCS nach dem Pool-Prinzip

Eine konkrete Art der OCS-Umsetzung ist das Pool-Prinzip. Der Pool beinhaltet alle Meldungen. Sie schwirren da sozusagen frei herum. Diese Meldungen beinhalten Daten, welche für eine bestimmte Zellenart oder einer

Gruppe von Zellen definiert sind.

Damit nun jeder Zellenhaufen an diese Meldungen gelangen kann, existiert einen globalen Zugriffskanal auf den Pool. Durch diesen Kanal kann jede Zelle Meldungen absetzen aber auch Meldungen anfordern und lesen. Eine Zelle wählt eine beliebige Meldungen im Pool aus. Kann sie die Meldung interpretieren, reagiert sie entsprechend darauf und hat somit diese Meldung verbraucht. Andere Zellen können auf diese Meldung nicht mehr reagieren. Kann eine Zelle die Botschaft nicht interpretieren, gelangt sie unverändert zurück in den Pool. Wie jede Zelle hat auch die Meldung eine Lebensdauer. Wird sie nicht gelesen oder niemand kann sie interpretieren, wird sie nach einer gewissen Zeit von Fresszellen gefressen, und die Ressourcen werden freigegeben.

Der Code der Meldung legt fest, welche Zellenart sie verbrauchen darf. Aber welche der Zellen gerade damit beschäftigt ist die Meldung wirklich zu verarbeiten, lässt sich nicht vorhersagen.

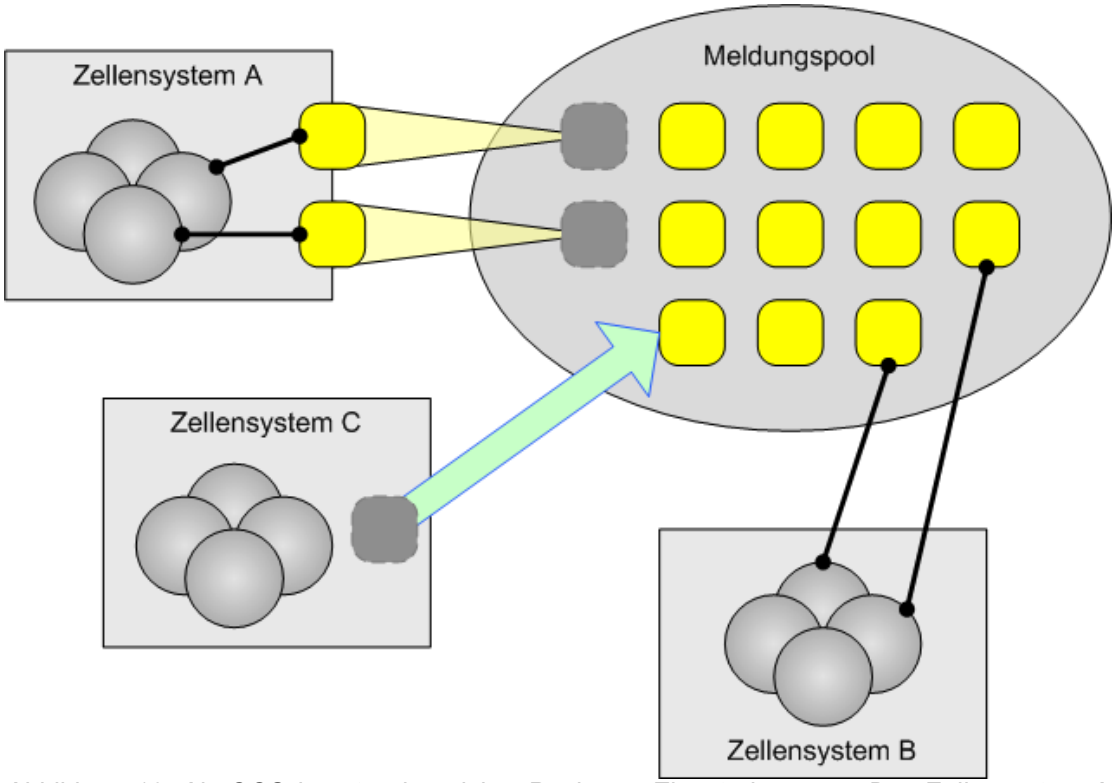

Abbildung 16: Als OCS könnte ein solcher Pool zum Einsatz kommen. Das Zellensystem A entfernt Meldungen zum Verarbeiten. Das Zellensystem B prüft, ob es die Meldung gebrauchen kann und das Zellensystem C sendet Meldungen an den Pool.

Ein Nachteil des Pools ist, dass es sich hier um eine zentrale und somit verletzliche Komponente handelt. Zentrale Objekte sind immer ein gutes

Angriffsziel von Viren und Hacker.

7.3 Beispiel: Hacker Angriff

Das folgendes Beispiel soll die Funktionsweise des OCS verdeutlichen:

Durch einen Angriff von aussen (durch Hacker oder Viren) wurden einige Zellen zerstört, auch drangen Virenzellen in das System ein, um weiteren Schaden zu verursachen. Die Sicherheitszellen haben den Angriff erkannt und sind nun beschäftigt, die Virenzellen zu eliminieren. Durch Botschaften an die Zellenfabrik, welche die Sicherheitszellen herstellt, wird bekannt gegeben, dass mehr Sicherheitszellen zu produzieren sind. Die Fabrik reagiert auf diese Botschaften und produziert nun vermehrt Sicherheitszellen. Durch den Angriff mangelt es im System an Arbeitszellen. Das System reagiert mit der Produktion dieser Zellenart auf die erhöhte Nachfrage. Während die zerstörten Zellen nun von den Fresszellen entfernt werden, übernehmen die umliegenden und neu produzierten Zellen die Produktion.

8 Arbeitsweise der Zellen

Wie arbeiten nun diese Zellensysteme zusammen, um etwas brauchbares herstellen zu können? Um dieser Frage auf den Grund zu gehen müssen wir die zu verrichtende Arbeit etwas genauer anschauen. Jeder Auftrag, jede Arbeit, jedes Problem kann in kleinere Einheiten aufgeteilt werden. Diesem Prinzip folgt jede Programmierung. Der Unterschied besteht aber in der Art und Weise der Aufteilung. Das CCM verlangt ein paralleles Denken. Mit anderen Worten, das Aufteilen der Arbeit geschieht nach dem Grundsatz: welche Teilbereiche können parallel abgearbeitet werden. Diese Teilbereiche müssen nicht einmal unabhängig von einander lösbar sein. Zwischen den parallelen Abläufen kann eine Kommunikation stattfinden, die als Synchronisation gebraucht werden kann.

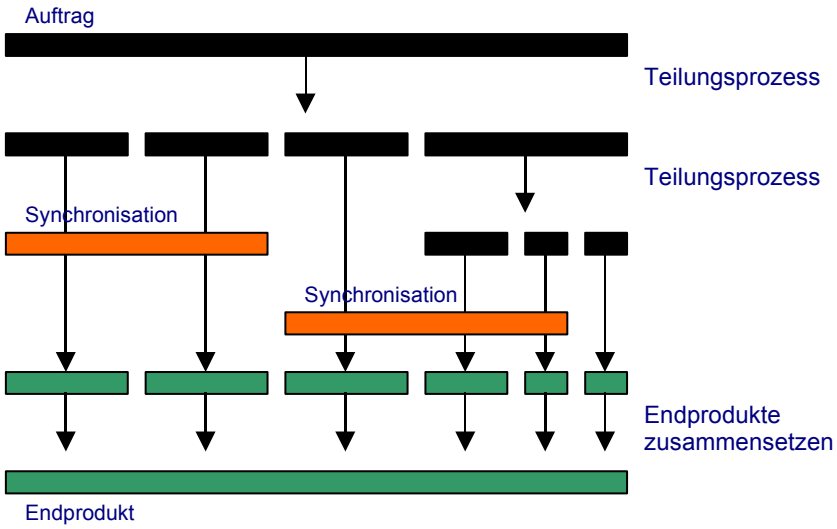

Abbildung 17: Aufteilen von Arbeitspaketen, paralleles Verarbeiten der kleinen Pakete und zusammenfügen der Resultate.

Der grosse Vorteil des parallelen Verarbeiten der Aufträge ist die enorme Zeitersparnis gegenüber dem linearen Vorgehen. Durch den Einsatz mehrerer Zellen ist der Auftrag ziemlich breit abgestützt. Das Aussetzen einer Zelle bedingt deshalb nicht einen kompletten Neubeginn, um den Auftrag zu erledigen. Eine neue Zelle führt die von der defekten Zelle hinterlassenen Arbeit zu Ende.

8.1 Aufteilen der Arbeit

Jeder Auftrag muss eine klar definierte Struktur aufweisen. Diese wird verwendet, um diesen Auftrag ordnungsgemäss aufteilen zu können. Die Aufteilung ist der erste Arbeitsschritt und wird von einer Zelle im System vorgenommen.

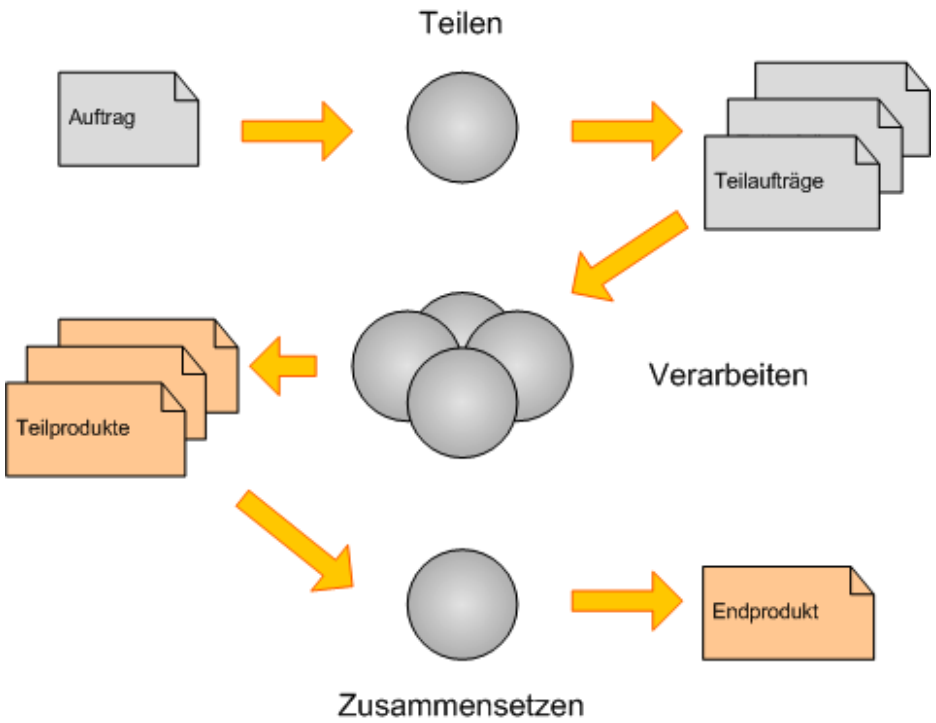

Abbildung 18: Das Aufteilen und Zusammenfügen der Arbeitspakete wird durch das Zellensystem selber vorgenommen.

Jede Zelle kann ihren Auftrag teilen, um so die Belastung herabzusetzen, indem sie Bearbeitung der Teilaufträge delegiert. Andere Zellen, sofern vorhanden, übernehmen dann diese Teilaufträge. Aus jeder Verarbeitung resultiert ein Produkt. Die Teilprodukte werden durch die Zelle, welche den Auftrag geteilt hat, wieder zusammengesetzt. Am Ende steht dann das Endprodukt. Es gibt natürlich beliebig viele Zwischenkombinationen: Ein Produkt kann zugleich ein Auftrag sein, welcher wieder von Zellen geteilt und verarbeitet werden kann.

Der Ablauf jedes Verarbeitungsprozesses muss genau geplant und nach diesem Schema umgesetzt werden. Dies kann mit Hilfe von Workflows und Petrinetzen erfolgen.

8.2 Beschreiben eines Workflows

Für jedes Arbeitspaket existiert eine genaue Verarbeitungsbeschreibung, der Workflow. Ein Workflow definiert im Cell Computing Model folgende Parameter:

- wo stehen wir in dem Verarbeitungsprozess
- müssen wir auf andere Teilprodukte warten
- an welche Zellen oder Zellensysteme muss das verarbeitete Produkt weitergesendet werden.

Um ein Workflow optimal und allgemein gültig zu beschreiben, verwenden wir Petrinetze. Petrinetze haben den grossen Vorteil, dass sie mathematisch analysiert werden können. Somit kann festgestellt werden, ob das Netz verklemmungsfrei und sicher ist.

Die Abbildung 19 zeigt ein Zellensystem A welches 4 Arbeitspakete zur Auslieferung besitzt. Die Transition (Quadrat) ist dann aktiviert, wenn die Bedingung erfüllt ist, dass heisst es müssen Vor- und Nachbereich der Transition erfüllt sein.

Eine Bedingung kann folgendes Enthalten:

- Hat es Arbeitspakete vom Typ X im Vorbereich (Zelle A). (Nach Definition vom Petrinetz)
- Hat es genügend Kapazität im Nachbereich. (Nach Definition vom Petrinetz)
- Ist das Zellensystem B erreichbar.

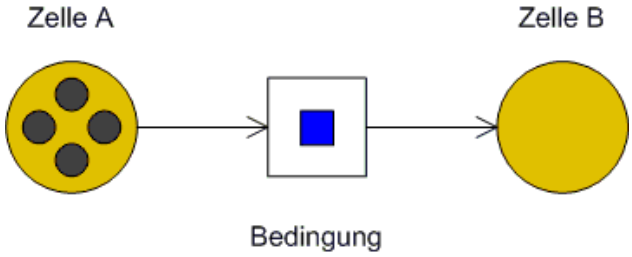

Abbildung 19: Darstellung eines Verarbeitungsprozesses mit 2 Zellensystemen

Ist die Bedingung wahr, wird das Arbeitspaket von der Zelle A an die Zelle B gesendet (Aktion).

8.2.1 Definieren von Bedingungen

Die obige Definition hat gewisse Nachteile. Wir müssen den Vorbereich kontaktieren und prüfen. Durch folgende Definition kann das Prüfen vereinfacht werden und den Ablauf schneller erfolgen:

- Fertig geleistete Arbeit wird von der Zelle A weiter geschickt. Dabei wird nur die Erreichbarkeit der Zelle B geprüft.
- Kapazität beim Zielsystem ist unendlich gross.

Zusätzlich können wir nun definieren, welche Aufträge (Tokens) eine Transition aktivieren. Dadurch können wir von einer Zelle aus definieren, dass unterschiedliche Arbeitspakete an verschiedene Zellen gesendet werden (Abbildung 20).

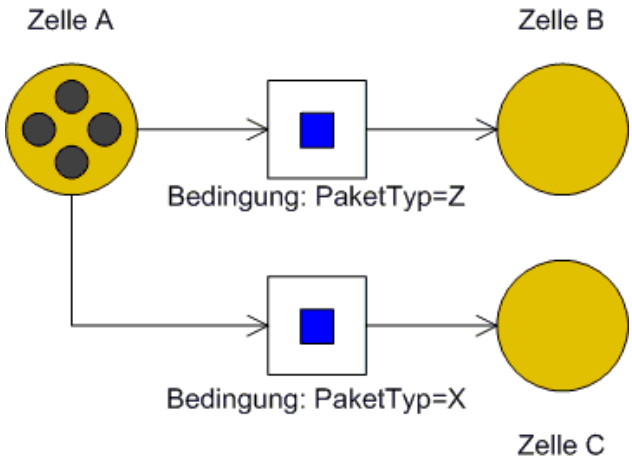

Abbildung 20: Es werden gleichzeitig die Pakete vom Typ X und Z an die entsprechenden Zellensysteme gesendet.

8.2.2 Synchronisieren

Ein wichtiger Bestandteil der Petrinetze ist das Synchronisieren und Parallelisieren von Abläufen. Das Synchronisieren wird gebraucht um

Teilprodukte zusammenzusetzen.

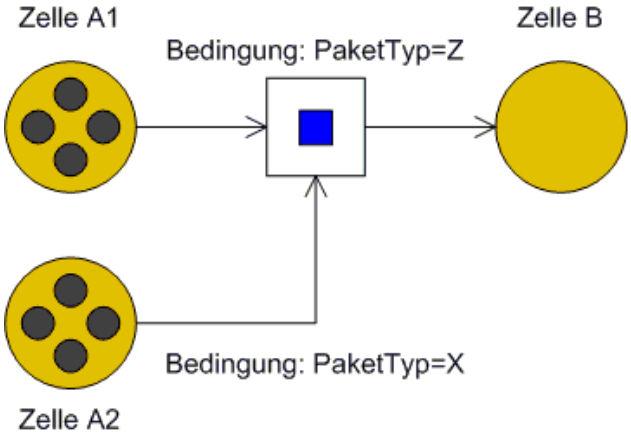

Abbildung 21: Synchronisieren der Zelle A1 und A2.

Die Zellen A1 und A2 (Abbildung 21) senden ihre Arbeitspakete an die Zelle B, welche diese Pakete zu einem Produkt zusammenfügt.

Die gesamte Koordination beim Synchronisieren ist nicht trivial lösbar. Hier ist ein möglicher Ansatz gegeben.

8.2.2.1 Das Synchronisationsverfahren

Da ein Netzwerk nicht als sicher eingestuft werden kann, muss eine Fehlerbehandlung eingebaut werden. Zu den obigen Punkten können verscheiden Strategien verwendet werden. Nehmen wir an die Zelle A1 ist bereit zur Auslieferung des Arbeitspakets Z.

1. Die Zelle A1 informiert die Zelle A2 mit einem Request R1, dass sie zur Auslieferung bereit ist.

   a) Die Zelle A2 kann nicht erreicht werden. Nach x-Versuchen abbrechen und durch Back Tracking den Fehler an die vorhergehenden Zellen zurückgegeben, oder den Fehler an eine Fehlerbehandlungsroutine weiterleiten.

2. Sobald die Zelle A2 bereit ist, wird ebenfalls einen Request R2 an die Zelle A1 gesendet.

   a) Die Zelle A1 ist nicht erreichbar. Gleiche Fehlerbehandlung wie bei Punkt 1.

   b) Die Zelle A1 ist erreichbar, das Arbeitspaket wurde aber verworfen, da

die Zeit abgelaufen ist (expired). Die Zelle A2 muss nun ihr Arbeitspaket auch verwerfen, da es nutzlos geworden ist.

3. Steht bei einer Zelle einen Request an, und die Zelle hat selber einen Request gesendet, wird eine Answer gesendet und auf die Answer der anderen Zelle gewartet.

   a) Request wurde gesendet, es kommt aber kein Answer auf diesen Request. Nach x-Versuchen oder nach einer definierten Zeit wird abgebrochen und die Fehlerbehandlung eingeleitet.

   b) Beim Senden der Answer ist die andere Zelle nicht mehr erreichbar. Es wird abgebrochen und die Fehlerbehandlung eingeleitet.

4. Beim Eintreffen der Answers werden die Arbeitspakete gesendet.

   a) Das Zielsystem ist nicht erreichbar. Fehlerbehandlung wird eingeleitet.

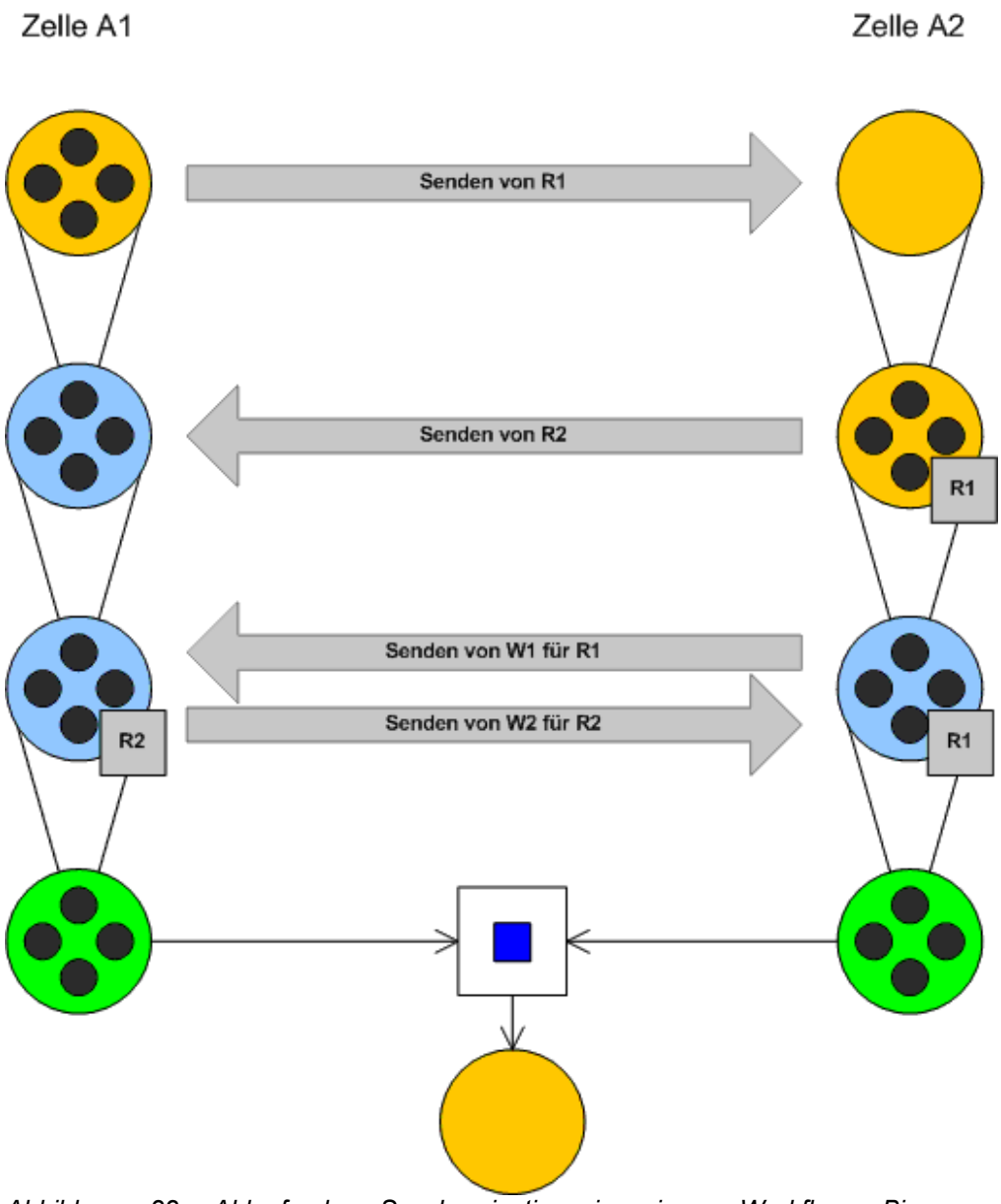

Abbildung 22: Ablauf der Synchronisation in einem Workflow. Bis zum Synchronisieren müssen die Zellen A1 und A2 untereinander ihre Bereitschaft bekannt geben.

8.2.3 Parallelisieren von Abläufen

Parallelisieren heisst, ein Arbeitspaket an zwei Zellensysteme versenden. Dabei wird das Arbeitspaket dupliziert.

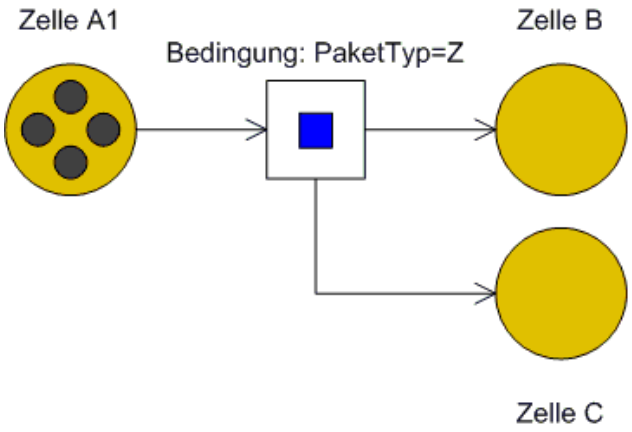

Abbildung 23: Parallelisieren von Abläufen

Dieser Ablauf ist wesentlich einfacher. Es müssen lediglich der Vorbereich und der Nachbereich geprüft werden.

8.2.4 Definieren von Workflows

Durch das Verwenden von Workflows, können wir im gleichen Netz verschiedene Aufträge verarbeiten, ohne dass wir das System neu konfigurieren müssen. Der Workflow wird mit jedem Arbeitspaket mit gegeben. Somit wissen die Zellen beim Erhalten des Auftrags, wie er bearbeitet werden muss.

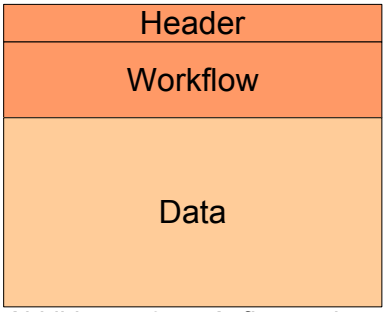

Abbildung 24: Aufbau eines Arbeitspaketes

8.3 Beispiel Versorgung mit Aufträgen

Wie gelangen Zellen an ihre Arbeit? Hier kann das DCS gebraucht werden. Jeder Auftrag hat eine Zieladresse. Diese Zieladresse definiert ein bestimmtes Zellensystem (Zellengruppe) für welche die Arbeit definiert ist. Das Zellensystem nimmt diesen Auftrag aus einem Kanal des DCS entgegen und nimmt eine gewisse Aufteilung der Arbeit vor (siehe oben). Die Teilaufgaben und Teilprodukte werden in einem internen Pool gespeichert. Jede Arbeitszelle hat einen speziellen Kanal auf diesen Pool, durch den sie die Aufträge entgegen nehmen kann. Jede Arbeitszelle prüften nun, ob ein Auftrag ansteht und nimmt diesen entgegen. Sobald der Auftrag erledigt ist, wird dieser wieder in den Datenpool des Zellensystems abgelegt. Dort wird der Auftrag entweder mit anderen Aufträgen wieder vereint oder an andere Zellen weitergeleitet. Dies geschieht nach dem Workflow. Er definiert wer, was und wie verarbeitet.

9 Kommandozentrale des Systems

Die Kommandozentrale ist die intelligente Komponente des CCM. Sie entspricht sozusagen dem Gehirn und übernimmt ähnliche Aufgaben, welche in diesem Abschnitt näher erklärt werden. Die Kommandozentrale ist noch am wenigsten beschrieben, ist aber eines der wichtigsten Komponenten des CCM. Durch sie wird das CCM erst zum System, welches autonome Fähigkeiten und künstliche Intelligenz besitzt.

9.1 Steuern der Zellen

Eine der Aufgaben ist das Steuern der einzelnen Zellensysteme. Wie bereits beschrieben, geschieht die Steuerung entweder durch direkte Kommunikation (Siehe DCS) oder durch die Trendkommunikation (Siehe OCS). Während die direkte Kommunikation dem Nervensystem bei biologischen Systemen entspricht, wird mit der Trendkommunikation eine Art Steuerung mit Hormonen realisiert.

Die Kommandozentrale empfängt Signale aus der Umgebung des Systems, wie Eingaben eines Benutzers oder Bilder von Kameras usw. Diese Signale werden von den Sensorzellen in Meldungen (Datenpakete) umgewandelt. Aus diesen Meldungen erfolgt der Start der Verarbeitung. Diese Signale werden meistens über die direkte Kommunikation erfolgen. Zuerst muss zu jeder Meldung oder Meldungsart eine geeignete Verarbeitungsbeschreibung (Workflow) hinzugefügt werden. Diese Beschreibung dient dem Verarbeiten, Aufteilen, Synchronisieren und dem Zusammenfügen mit anderen Daten. Die Verarbeitungsbeschreibung ist entweder fix codiert oder durch Erfahrung des Systems entstanden.

Über die Trendkommunikation können die Zellensysteme nachhaltig beeinflusst werden. Zum Beispiel eine Stresssituation könnte ein Trend auslösen, welcher Zellenfabriken veranlasst mehr Zellen zu produzieren. Das Feststellen, ob eine solche Situation vorliegt kann einerseits lokal erfolgen (in einem Zellensystem) oder durch globale Verhaltensmuster des Gesamtsystems. Während ersteres auch lokal gelöst werden kann, muss das Zweite durch die Kommandozentrale gelöst werden.

9.2 System Erfahrung

Eine weitere wichtige Aufgabe der Kommandozentrale ist, gewisse Entscheidungen selber fällen zu können und je nach Erfahrung selbständig Einfluss auf die jeweiligen betroffenen Zellensysteme zu nehmen. Die Kommandozentrale hat direkten Zugriff auf eine Erfahrungsdatenbank. Hier können die Erfahrungen miteinander verknüpft und gespeichert werden. Diese Erfahrungsdatenbank könnte zum Beispiel mit einem neuronalen Netzwerk implementiert werden, in dem jedes Neuron auch eine Zelle ist.

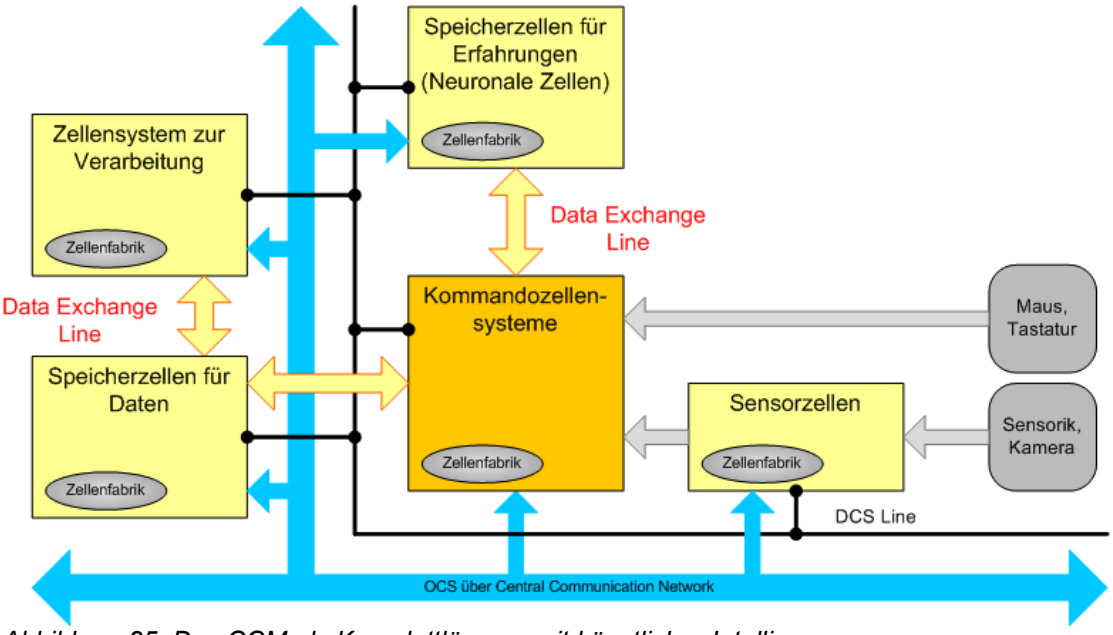

Abbildung 25: Das CCM als Komplettlösung, mit künstlicher Intelligenz

9.3 Evolutionäre Weiterentwicklung

Die Fähigkeit gewisse Zellen zu züchten und weiter zu entwickeln wird ebenfalls durch Bereiche in der Kommandozentrale gesteuert. Wie weit dieses Prinzip eingesetzt werden kann, um die DNA der jeweiligen Zellen zu verändern, bzw. gewisse Mutationen vorzunehmen, muss noch im Detail abgeklärt werden.

10 Die Datenablage

10.1 Lokale Zellendaten

Die lokalen Zellendaten dienen der Zelle als Arbeitsdaten. Sie werden nur von einer bestimmten Zellenart verwendet und können von anderen Zellenarten nicht ohne weiteres verwendet werden. Es handelt sich hier nur um temporäre Daten, welche von einem Zellensystem verarbeitet werden. Nicht jede einzelne Zelle hat einen lokalen Speicher. Der lokale Speicher wird vom Zellensystem verwaltet. Das hat den Vorteil, dass die Daten von verschiedenen Zellen gleichzeitig verarbeitet werden können. Kann eine Zelle nun die Arbeit nicht zu Ende führen, übernimmt eine inaktive Zelle die verbleibende Arbeit. Dieses Data Sharing verlangt aber eine klare Absprache zwischen den arbeitenden Zellen.

10.2 Die Speicherzellen

Im Gegensatz zu den lokalen Daten können Daten auch global in den Speicherzellen abgespeichert werden. Die Speicherzellen stellen eine globale Datenbank dar, in welcher jeder mit Zugriffsrecht zugreifen und Daten verarbeiten kann.

Es gibt zwei Hauptgruppen von Speicherzellen:

- Speicherzellen für Verarbeitungsdaten
- Speicherzellen für die Erinnerungswerte

Die Erinnerungswerte werden von der Kommandozentrale verwaltet. Sie entstehen durch die Lernfähigkeit und den Erfahrungen des Systems. Es entsteht eine netzartige Verknüpfung zwischen diesen Daten, eine Möglichkeit gewisse Daten mit anderen Daten in Verbindung zu bringen.

Die Speicherzellen der Verarbeitungsdaten könnten mit einer relationalen Datenbank implementiert werden.

10.3 Beispiel einer Datenverwaltung

Nehmen wir an, eine Kamera generiert Bilder. Diese Bilder werden nun von den Sensorzellen zuerst so umgewandelt, dass das System sie interpretieren kann

Real World -> Hardware -> Sensorzellen (Treiber) -> Bildformat (jpg, gif etc.)

Nun sollen diese Bilder von gewissen Zellen weiterverarbeitet werden. Damit dies geschehen kann, werden sie von den Sensoren über die Kommandozentrale in Speicherzellen abgelegt. Über die Kommandozentrale werden nun die entsprechenden Zellen animiert, diese Informationen zu verarbeiten. Die Bilder werden zum Teil kopiert und als lokale Daten in der jeweiligen Zellengruppe abgespeichert. Nach der Verarbeitung der Bilder werden diese wieder in den Speicherzellen abgelegt, und stehen nun anderen Zellensystemen zur Verfügung, die sie je nach dem weiterverarbeiten oder ausgeben.

10.4 Einige Zellenarten

Arbeitszellen:	Die Arbeitszellen dienen der Datenverarbeitung des Systems. Sie entsprechen der logischen Einheit des Systems.
Sensorzellen:	Diese Zellen entsprechen der Dateneingabe eines herkömmlichen Systems.
Ausgabezellen:	Diese Zellen dienen als Schnittstelle zwischen dem Benutzer und dem System. Sie geben Aufschluss, was das System verarbeitet hat (Ausgabe).
Fabrikzellen:	Hier werden je nach Bedarf verschiedene Zellenarten produziert. Die Fabrikzellen sind lebenswichtig für das System.
Fresszellen:	Diese Zellen räumen das System auf und entfernen verwaiste und zerstörte Zellen
Sicherheitszellen:	Zellen dieser Art sind für die Sicherheit zuständig und zerstören fremde und gefährliche Zellen.
Versorgung:	Diese Zellen transportieren Meldungen durchs

System (langsame Kommunikation oder Trendkommunikation)

Speicherzellen: Speicherzellen verwalten den Datenspeicher des Systems. Sie sind also die Schnittstelle zwischen dem System und der Datenablage.

Meldungszellen: Die Meldungszellen beinhalten die Aufträge mit der Auftragsbeschreibung (Code). Es sind sehr schlanke Zellen, die für den Transport zuständig sind.

11 Versionsgeschichte

Änderungen am Dokument sind hier festgehalten

Version	*Beschreibung*	*Datum*
0.1	Projektarbeit Cell Computing Model	26.06.05
0.2	Refactoring des Frameworks	09.09.05
0.3	Neues Strukturieren und Trennen von Implementation und Theorie	30.09.05
0.4	• Neue Graphiken • Mobile Zellen als Lösung für globale Sicherheit • Workflow Definition mit Petrinetzen	05.12.05
0.5	Abgabe der Diplomarbeit	23.12.05

Literaturverzeichnis

[GEOK2]: Diverse, GEO kompakt. Das Wunder Mensch, 2005

[MSENC]: Microsoft, Microsoft Encarta, 2001