Vorgelegt an der Alpen-Adria Universität Klagenfurt

Systemdenken, Selbstorganisation und Informationstechnologie? Exemplarisches Fallbeispiel: Bittorrent

Seminar aus Angewandter Informatik
Univ. Prof. Dr. Laszlo Böszörmenyi

von
Markus Echterhoff
Matrikelnr.: 0560785
Klagenfurt
12. Februar 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Abstract

Die exponentiell steigende Leistungsfähigkeit und hohe Verfügbarkeit moderner Computer erlaubt es zunehmend alternative, natur-inspirierte Algorithmen sinnvoll einzusetzen und natürliche Systeme in einer kontrollierten Umgebung zu erforschen. Außer der Leistungsfähigkeit unserer Rechenmaschinen benötigen wir jedoch ein tieferes Verständnis dieser Systeme, um erstgenannte Algorithmen zu programmieren, sowie ein gemeinsames Lexikon, um unsere Ideen zu kommunizieren. Wir müssen uns darüber einig werden, was wir unter den Begriffen *System*, *Struktur*, *Komplexität*, *Emergenz* oder *Kritikalität* verstehen und darüber hinaus unsere Sichtweise ändern. Statt eine Welt voller Systeme zu beobachten, müssen wir uns darüber klar werden, dass wir selbst System und Systemkomponente sind.
Diese Arbeit trägt grundlegende Definitionsversuche für ein gemeinsames Lexikon zusammen und verschafft einen Überblick über die Mechanismen, die Selbstorganisation zugrunde liegen oder eng mit ihr verwandt sind. Um der Nähe zur Natur dieses Themas gerecht zu werden, verwendet der Autor einen ganzheitlichen Problemzugang: Es werden Quellen aus verschiedenen Fachgebieten und interdisziplinärer Forschung verwenet, um für die aus ihrer Natur heraus zT. unscharfen Definitionen eine breite Basis für Verständnis zu schaffen, die auf der Prämisse "I know it, when I see it" (dt.: "Ich erkenne es, wenn ich es sehe") beruht und auch dort noch für gleiches Verständnis unter verschiedenen Lesern sorgt, wo die Definitionen unzureichend sind.
Das etablierte Wissen wird um die Bedeutung des Beboachters ergänzt und verwendet um, als stellvertretendes Beispiel, das Bittorrent-Protokoll systemisch zu analysieren, wobei sich herausstellt, dass dort gewonnenes Wissen übertragbar auf andere Systeme ist, was die Universalität des Systembegriffs unterstreicht.

2 Einleitung

Selbstorganisation

> "Geprägt wurde dieser Terminus in den 50er Jahren von den beiden Elektroingenieuren W.A. Clark und B.G. Farley. Sie erkannten, daß sich Operatoren, die in einer geschlossenen Beziehung stehen, irgendwie stabilisieren und beobachteten – noch ohne eine Theorie der rekursiven Funktionen oder des

> Eigenwertes zu kennen – das Phänomen, daß bestimmte geschlossene Systeme nach einer gewissen Zeit stabile Formen des Verhaltens entwickeln." [von Förster, 1998]

Dieses Zitat beschreibt die Geburt des Begriffes der Selbstorganisation und beleuchtet gleichzeitig ein Kernproblem der modernen Informationstechnologie sich diesem Thema anzunähern. Selbstorganisation ist dem obigen Zitat nach ein Phänomen, kein Konstrukt. Daraus folgt, dass Selbstorganisation nicht direkt erzeugt werden kann, sondern Modelle geschaffen und implementiert werden müssen, die jenen Systemen ähnlich sind, bei denen selbstorganisierendes Verhalten beobachtet werden kann. Solche Systeme wurden in Fachgebieten der Natur- und Geisteswissenschaften unabhängig voneinander beobachtet und nur zum Teil mithilfe der Mathematik beschrieben. Nach erfolgter Modellierung muss das Modell untersucht werden, ob es selbstorganisierende Eigenschaften aufweist, oder ob sie in der Modellierung verloren gegangen sind.
Um dies beurteilen zu können, muss der Beobachter ein Verständnis davon haben, was die Charakteristika selbstorganisierender Systeme sind. Diese Arbeit trägt solche Charakteristika zusammen, um den Leser in die Lage zu versetzen, selbstorganisierendes Verhalten zu erkennen und zu kommunizieren. Da der Autor der vorliegenden Arbeit kein Freund von langen Definitionslisten ist, werden Definitionen unmittelbar dort eingeführt, wo sie gebraucht werden. Diese Arbeit hat das große Glück, sich dem Leser oft durch Beispiele verständlich machen zu können, denn viele der Konzepte der Theorie der Selbstorganisierung verstecken sich in ganz alltäglichen Dingen. Es müssen also nicht vollständig neue Konzepte erlernt, sondern lediglich erkannt und übertragen werden. Die verwendeten Beispiele sind vorzugsweise aus dem Fachgebiet der Informationstechnologie ausgewählt, da dem Leser ein Hintergrund in dieser Domäne unterstellt wird. Wo das Illustrationsvermögen von Beispielen aus der IT denen anderer Disziplinen unterliegt, werden die demonstrativeren Beispiele bevorzugt. Diese Arbeit sieht ihre erste Pflicht darin, dem Leser die Konzepte zu verdeutlichen, mit denen er selbstorganisierende Systeme identifizieren und klassifizieren kann.
Auf Basis des so etablierten Wissens wird als stellvertretendes Beispiel aus der Informationstechnologie das Bittorrent-Protokoll und seine Anwendung im Hinblick auf selbstorganisierendes Verhalten untersucht.
Dem Leser wird ein Grundverständnis des Englischen unterstellt, deshalb wird auf Übersetzungen von Zitaten mit einem oder wenigen Sätzen verzichtet.

3 Systeme

Was hat ein Betriebssystem mit einem Schimpansen gemeinsam?

Für diese Arbeit sind sich diese zwei Dinge sehr ähnlich in der Weise, dass sie beide als System aufgefasst werden können und sehr unterschiedlich in der Art, wie sie den Systembegriff implementieren. Ein Betriebssystem ist (derzeit) ein statisches, mechanisches System, ein Schimpanse ein autopoietisches (siehe entsprechendes Kapitel) System, das sich evolutionär entwickeln kann. Die unterschiedlichen Eigenschaften und Besonderheiten dieser Systeme wird der Leser in den folgenden Kapiteln kennen lernen.

Bevor wir auf den selbstorganisierenden Charakter bestimmter Systeme eingehen, scheint es unumgänglich zu definieren, was unter dem Begriff *System* zu verstehen ist. Sprachlich leitet sich das Wort aus dem Griechischen her, wo "Systema" soviel bedeutet wie etwa "Zusammenstellung".

Definition 1 *"A system is a set of objects, together with relationships between the objects and between their attributes." [Hall and Fagen, 1956]*

Eine weitere Definition, die mehr Wert auf einen abgeschlossenen Charakter legt und die Attribute der Teile des Systems vernachlässigt, ist die Folgende:

Definition 2 *"Initially we can define a system broadly and crudely as any entity, conceptual or physical, which consists of interdependent parts." [Ackoff, 1961] nach*

Im Folgenden bezeichnen wir die Objekte oder Teile eines Systems auch als Komponenten. Diese Definition besagt also, dass die Komponenten Teil einer Einheit sind. Eine Einheit im Sinne von "Eins sein" zieht ein Abgrenzen gegenüber "dem Anderen", das eben nicht die Einheit ist, nach sich. Dieses Andere wird im Folgenden als *Umgebung* des Systems bezeichnet. Wir halten als Definition fest:

Definition 3 *Die Umgebung eines Systems sei alles, was nicht als das System selbst erkannt wird.*

Mit dieser Umgebungsdefinition führen wir implizit eine erkennende Instanz ein, die in den beiden vorhergegangenen Definitionen des Systembegriffs vernachlässigt wurden. Wir gehen im nächsten Kapitel näher auf die Rolle des Beobachters ein. Jetzt, wo wir System und Umgebung definiert haben, benötigen wir noch den Begriff der *Systemgrenze*.

Definition 4 *Die Grenze eines Systems sei der Übergang von System zu Umgebung.*

Man beachte, dass sich topologische Systemgrenzen, wie etwa eine Zellmembran, idR. nicht mit konzeptuellen Systemgrenzen decken, wenn ein System die Betrachtung von beidem zulässt. So macht es zB. in der Gruppendynamik keinen Sinn als die Grenze der Gruppe die räumliche Anordnung ihrer Mitglieder zu betrachten. Vielmehr wird die Grenze hier so betrachtet:

> "Man könnte sagen, dass jede Gruppe – wie auch jedes Individuum – über eine Identität verfügt, allerdings über eine kollektive Identität („Wir-Gefühl"): Man fühlt sich zugehörig zu einer bestimmten Gemeinschaft (Freundeskreis, Projektgruppe, Arbeitsgruppen usw.), mit eigenen, mehr oder weniger bewussten Normen und Gesetzen, gewissermaßen umschlungen von einer „sozialen Haut", welche als Grenze der Gruppe einzelne Personen als zugehörig oder nicht zugehörig definiert." [Krainz, 2006]

Bezüglich der Systemgrenze, ob konzeptuell oder raum-zeitlich, lässt sich zwischen *offenen* und *abgeschlossenen* Systemen unterscheiden.

Definition 5 *"Hinsichtlich seiner Umweltbeziehungen bezeichnet man ein System als offen, wenn es mit seiner Umwelt Austausch pflegt, wobei neben Materie und Energie vor allem auch Informationsaustausch in Frage kommt, und wenn es gegenüber Neuem ... offensteht. Systeme ohne Austausch mit der Umwelt nennt man abgeschlossene oder isolierte Systeme." [Jantsch, 1979]*

Es existiert die feinere Unterscheidung zwischen *operationaler* und *energetischer* Offenheit [Stein and Varela, 1993] (bzw. analog Abgeschlossenheit). Die vorhergegangene Definition der Offenheit eines Systems entspricht hier der energetischen Offenheit (Die Materie oder Informationen mit einschließt). Operationale Offenheit beschreibt, ob die Operationen des Systems die Systemgrenzen überschreiten können (*operational offen*), oder ob das System nur als ganzes mit seiner Umwelt interagiert (*operational geschlossen*).

3.1 Man schaut mit den Fingern, nicht mit den Augen

Beide bisherigen Definitionen des Begriffs des Systems vernachlässigen den Beobachter, der für die Analyse eines Systems jedoch von beachtlicher Bedeutung ist. Erkennen ist ein aktiver Prozess, eine Handlung also, die etwas erzeugt. Die Qualitäten des Erzeugnisses liegen in der Struktur des erkennenden Systems begründet, nicht im Erkannten. Maturana und Varela etablieren diesen Umstand sehr ausführlich und fassen zusammen: "..., daß jeder Akt des Erkennens eine Welt hervorbringt." [Maturana and Varela, 1987] Jack Kornfield illustriert sehr eingängig eine psychologische Sichtweise auf die Problematik des Erkennens: "If we are hungry and we walk down the street, we don't see shoe stores or the weather or the clouds. We see there is a nice Greek restaurant." [Kornfield, 2001]
Abbildung 1 visualisiert die Kreativität des Erkennens bezüglich der Bestimmung von Grenzen in einem System: In einem Software-Programm könne man die Topologie des Netzwerkes, in dem sich die betreffenden Computer befinden, herausfinden. Obendrein seien die Endgeräte des Netzwerks bekannt und eindeutig einer Position im Netzwerk zuordenbar. Ein Software-Entwickler A stelle sich das links abgebildte Netzwerk

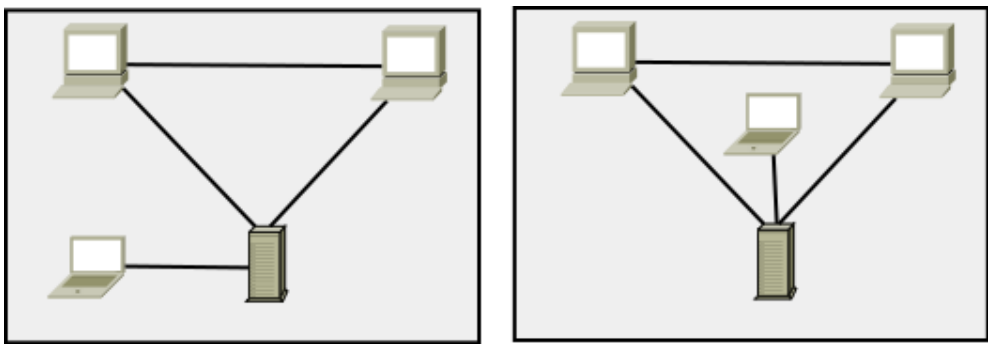

Abbildung 1: **Grenzen eines Computernetzwerks**

vor und ein Entwickler B das rechts abgebildte. Da die Netzwerktopologie nichts mit der physischen Topologie der Endgeräte zutun hat, aber die Entwickler in Endgeräten denken, sehen beide Entwickler unterschiedliche Systemgrenzen. Im Speziellen sieht B den Laptop uU. nicht als Teil der Grenze.
Um den Beobachter eines Systems zu berücksichtigen, sei hier eine dritte Definition für den Systembegriff genannt, die zusammen mit den beiden vorhergegangenen ein ausreichendes Begriffsverständnis bilden soll. Bemerkenswert an dieser Definition ist, dass sie in einem Nebensatz auftaucht, quasi als kurze Begriffsklärung.

Definition 6 *"... ein System, d.i. ein nach Prinzipien geordnetes Ganzes der Erkenntnis ..." [Kant, 1786]*

Aus diesen drei Definitionen setzt sich ein Verständnis zusammen, das einen Beobachter vorraussetzt, der eine Einheit erkennt. Die als System erkannte Einheit besteht aus Komponenten, die miteinander in einer Abhängigkeitsbeziehung stehen und deren Attribute ebenfalls in Beziehung stehen können. Diese Beziehungen folgen bestimmten Prinzipien. Man beachte, dass die Systemkomponenten wiederum Systeme sein können. Vorraussetzung ist ein Betrachter, der dies durch die rekursive Anwendung dieses Systemverständnisses erkennt.

3.2 Emergenz

> "Eine Blume ist keine Blume. Darum ist sie wahrhaftig eine Blume."

Dies ist die Dialektik des Diamant-Sutras [Hanh, 1999], einer buddhistischen Lehrschrift. Gemeint ist damit, dass eine Blume aus Teilen besteht, von denen keines die Eigenschaft einer Blume aufweist. Eine Blüte ist genausowenig eine Blume, wie ein Stengel, Blätter oder Wurzel. Aber führt man all diese Komponenten zusammen, erhält man (wahrhaftig) eine Blume.
Wir beginnen mit diesem Beispiel um uns zu verdeutlichen, dass man Emergenz auch dekomposiv betrachten kann, was die entgegengesetzte Richtung der üblichen Beschreibung nach [Aristoteles,] ist:

> "Das was aus Bestandteilen so zusammengesetzt ist, dass es ein einheitliches Ganzes bildet, ist nicht nach Art eines Haufens, sondern wie eine Silbe, das ist offenbar mehr als bloss die Summe seiner Bestandteile. Eine Silbe ist nicht die Summe ihrer Laute: ba ist nicht dasselbe wie b plus a, und Fleisch ist nicht dasselbe wie Feuer plus Erde."

Der Autor zweifelt im Übrigen daran, dass Aristoteles tatsächlich "mehr" meinte, denn offensichtlich können auch Eigenschaften der einzelnen Komponenten verloren gehen (Wie zB. ihre Bindungsfähigkeit), daher liegt nahe, dass es ursprünglich "anders" hieß und in der Übersetzung als "mehr" verloren ging, da "mehr", von einem naiven Standpunkt gesehen, logischer erscheint.
Wie wir später bei der Analyse des Bittorent-Protokolls sehen werden, können wir derartig emergente Systemeigenschaften nutzen. So kann dort uU. eine Datei vollständig im Netzwerk sein, aber auf keinem der teilnehmenden Computer vollständig vorliegen. Die Existenz dieser Datei ist also eine Eigenschaft des Gesamtsystems, nicht der Komponenten.
Emergenz hat auch überraschenden Charakter, genau dann, wenn die Systemeigenschaft bei Erzeugung des Systems nicht vorausgesehen wird. Aufgrund der komplexen Natur mancher Systeme, ist dies nicht immer möglich die Systementwicklung vorrauszusagen oder sie zu nützen; so werden wir zB. trotz aller Vorhersagen immernoch vom Platzregen erwischt, wenn doch eigentlich die Sonne scheinen sollte. Man könnte sich auch fragen, ob Bewusstsein nicht mehr als ein emergentes Phänomen des komplexen Wirkens im menschlichen Gehirn ist. Festzuhalten ist hier, dass die Beobachtungsfelder von Emergenz mannigfaltig sind.

3.3 Organisation und Struktur

Moderne Computer verfügen häufig über redundante Speichersysteme wie zB. ein RAID (Redundant Array of Inexpensive Disks) vom Typ 1, in dem mehrere Festplatten zu einer Einheit zusammengestellt sind und als solche addressiert werden. Wir betrachten den Festplattenverbund als System, bei dem Information, die das System betritt auf allen Platten redundant abgelegt wird. Ein Benutzer habe ausschließlich magnetische Festplatten verbaut. Nun falle eine der Platten aus und er ersetze sie durch eine moderne SSD (Solid State Disk). (Bevor dem Leser sein Domänenwissen durch lauten Protest vom Verstehen der Aussage dieses Beispiels abhält, sei angenommen, dass der Benutzer plane, nach und

nach, alle Platten durch solche mit SSD Technologie zu ersetzen.) Das RAID spielt die Daten automatisch von den funktionierenden Platten auf die neue.
Die Frage, die wir uns stellen wollen, ist, ob es sich um das gleiche System handelt oder nicht, denn eine der Komponenten ist durch eine andere ersetzt worden. Was, wenn wir alle Festplatten durch ihre SSD-Äquivalente ersetzen?
Um die Frage zu beantworten, führen wir die Begriffe der Organisation und der Struktur eines Systems ein.

Definition 7 *"Unter Organisation sind die Relationen zu verstehen, die zwischen den Bestandteilen von etwas gegeben sein müssen, damit es als Mitglied einer bestimmten Klasse erkannt wird." [Maturana and Varela, 1987]*

Definition 8 *"Unter der Struktur von etwas werden die Bestandteile und die Relationen verstanden, die in konkreter Weise eine bestimmte Einheit konstituieren und ihre Organisation verwirklichen." [Maturana and Varela, 1987]*

Die Antwort auf unsere Frage lautet also, dass die Organisation des RAIDs die gleiche bleibt, aber die Struktur sich geändert hat.

4 Dynamik, Komplexität und Chaos

Diese Arbeit behandelt selbstorganisierende Systeme; solche Systeme sind dynamisch, im Gegensatz zu statischen Systemen. Dynamisch sein bedeutet sich in der Zeit zu entwickeln, was sich, gemäß [Leven et al., 1989], durch Rekursion ausdrücken lässt.

$$x(t+1) = f(x(t)), t = 0, 1, 2, ... \tag{1}$$

x zu einem beliebigen Zeitpunkt t ist also das sukzessive Ergebnis der Operation $f()$ rekursiv angewendet auf sämtliche x zu vorherigen Zeitpunkten $t - \Delta t$ bis zum Beginn des Systems $t = 0$.
In dynamischen Systemen sind die Einzelschritte $f()$ nicht umkehrbar und der Gesamtprozess bewegt sich entweder auf einen strukturellen Gleichgewichtszustand hin, oder zirkuliert fern vom Gleichgewicht. [Jantsch,

1979] Wir werden auf diese Unterscheidung noch näher eingehen, wenn wir Selbstorganisation untersuchen.

Wann bezeichnet man ein System als komplex, wann als chaotisch?

Definition 9 *"We use the term complexity to denote the existence of system properties that make it difficult to describe the semantics of a system's overall behavior in an arbitrary language, even if complete information about its components and interactions is known." [Bar-Yam, 1997] nach [Meer and Koppen, 2005]*

Ein komplexes System ist also ein System, das schwer zu beschreiben ist, selbst wenn man alles (bzw. "viel", wenn man selbstkritisch bleibt) darüber weiß. Ein Beispiel sind zB. Finanzmärkte.
Zu den komplexen Systemen zählen auch die chaotischen Systeme. Die Existenz chaotischer Systeme ist spätestens seit dem Schmetterlingsgleichnis (*butterfly-effect*) allgemein bekannt. Kann der Flügelschlag eines Schmetterlings andernorts einen Wirbelsturm auslösen? Es ist zu beachten, dass hierbei das Augenmerk nicht etwa auf einer Kettenreaktion liegt, sondern darauf, dass man das Entstehen eines Wirbelsturms auf Anfangsbedingungen zurückführen muss, bei denen sich scheinbar insignifikante, in der klassischen, mechanischen Physik gern als *vernachlässigbar klein* bezeichnete Größen, aufgrund der Dynamik des Wettersystems signifikant auswirken (oder auch vom System gedämpft werden und untergehen) können.

Definition 10 *"...sehr kleine Änderungen in den Anfangsbedingungen bewirken große Unterschiede im Endzustand, und da Zustände nur mit endlicher Genauigkeit gemessen werden können, sind somit der Voraussagbarkeit Grenzen gesetzt. Solche Systeme werden heute chaotisch genannt." [Leven et al., 1989]*

Programmierer wissen um das Problem der Genauigkeit nur zu gut. So ist es in der Praxis oft ratsam, Logarithmen zu addieren, statt zu Potenzieren um zu vermeiden, dass die Ungenauigkeiten von Gleitkommazahlen in den Vorkommabereich propagieren. Dynamische, komplexe Systeme wurden mit der Entwicklung von rechenstarken Computern ein zugänglicheres Gebiet, da die Systeme mithilfe von Computern numerisch ent-

wickelt werden konnten, was zuvor einen unüberwindbar großen Rechenaufwand darstellte. Die Verwendung von Computern im Bezug auf chaotische Systeme ist allerdings, durch die zuvor genannte Ungenauigkeit von Gleitkommazahlen, nur begrenzt möglich. Dedizierte Anwendungen verwenden alternative Darstellungen, wie Stringrepräsentationen für extrem große Zahlen, oder Zähler-Nenner Darstellung für rationale Zahlen. Numerische Berechnungen chaotischer Systeme, die nicht-rationale, reelle Zahlen benötigen, können durch Berechnungen mit dem Computer nicht akkurat entwickelt werden, da von $\sqrt{(2)}$ zB. idR. nur die ersten 64 Bit an Information gespeichert werden. Die durch das Abschneiden entstandene Ungenauigkeit kann in einem chaotischen System der Flügelschlag sein, der dazu beiträgt, dass sich das System in der Simulation anders entwickelt als es in der Realität würde.

5 Selbstorganisierende Systeme

Von den klassischen Systemen setzen sich selbstorganisierende Systeme entsprechend dem Zitat aus der Einleitung dadurch ab, dass ihre Struktur von den Systemeigenschaften selbst bestimmt wird und dass man ihre Struktur nicht nur räumlich, sondern auch zeitlich beschreiben muss, da sie sich in der Zeit entwickeln. [Jantsch, 1979] Selbstorganisation ist unterscheidbar in konservative Selbstorganisation und dissipative Selbstorganisation.

5.1 Konventionelle Selbstorganisation

Systeme, die konventionelle Selbstorganisation aufweisen, entwickeln sich in der Zeit auf ein strukturelles Gleichgewicht (Equilibrium) zu. Diese Entwicklung wird auch als *Devolution* bezeichnet. Die Entwicklungsrichtung kann nicht umgekehrt werden, dh. Änderungen, die solche Systeme an sich vornehmen, können nicht rückgängig gemacht werden. Ein Wachstum des Systems ist möglich. [Jantsch, 1979] Abbildung 2 zeigt den Devolutionsvorgang eines abgeschlossenen Systems in evoPaint, einem Softwareprojekt des Autors dieser Arbeit. Jeder Pixel repräsentiert einen probablistischen, zellulären Automaten. Zu Beginn sind sämtliche Pixel des Bildes mit zufälligen Farbwerten belegt. Das Zusammenwirken von hier 10.000 Automaten einer entsprechenden Konfiguration devolviert die Struktur des Systems in ein Equilibrium, das in unter 2000 Zeiteinheiten (Eine Zeiteinheit entspricht einer Sequenz der Handlungen

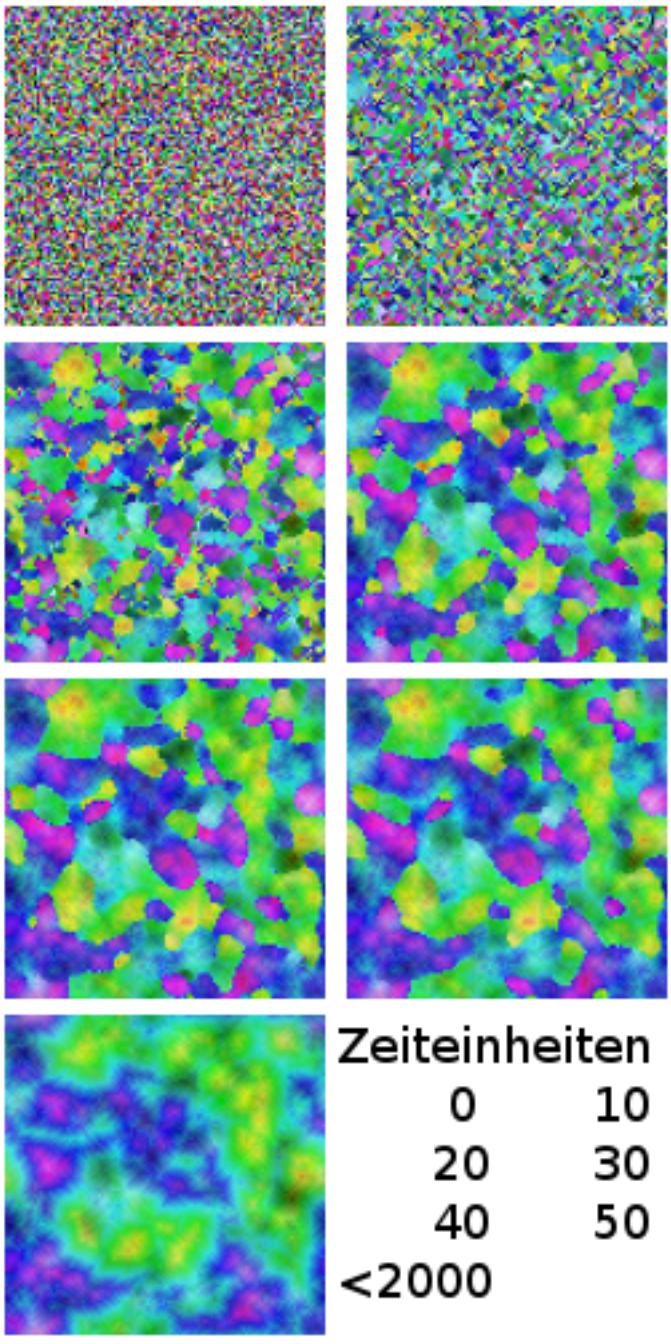

Abbildung 2: **Visualisierung einer Devolution in evoPaint**

aller Automaten) erreicht wird.
Konventionelle Selbstorganisation ist in der informationstechnischen Praxis häufig das gewünschte Systemverhalten, da es durch die Selbstordnung eine Strukturbeschreibung nach endlicher Zeit zulässt, die die Kommunikation über das System vereinfacht und anhand derer einfache Effizienzüberlegungen durchgeführt werden können. Besonders beliebt ist hierbei die Struktur der *Hierarchie*, wie man sie in ihrer abstraktesten Form in einem Wurzelbaum wiederfinden kann.

5.2 Dissipative Selbstorganisation

Definition 11 *"...ständig Entropieproduktion aufrechterhalten, also ständig »arbeiten« und Energie umsetzen. Man spricht in diesem Falle von dissipativer Selbstorganisation im Gegensatz zu konservativer Selbstorganisation, bei der nur die anziehenden und abstoßenden Kräfte im System selbst eine Rolle spielen." [Jantsch, 1979]*

Dissipativ selbstorganisierende Systeme stehen im Austausch mit ihrer Umwelt, sind also *offene* Systeme und befinden sich in einem Zustand fern vom Gleichgewicht, auf das sich die konservative Selbstorganisation hinentwickelt. Weiter besitzen solche Systeme *katalytische Prozesse*. Katalytische Prozesse sind solche, in denen bestimmte Komponenten an Operationen teilnehmen, in denen sie direkt (*Autokatalyse*), oder indirekt (*Crosskatalyse*) für die Bildung von Komponenten ihrer eigenen Art benötigt werden. Die für dissipative Selbstorganisation interessanten Systeme lassen sich häufig durch, von Manfred Eigen so benannte, *Hyperzyklen* beschreiben. [Jantsch, 1979]

5.2.1 Hyperzyklen

Definition 12 *"Ein Hyperzyklus ist ein geschlossener Kreis von Umwandlungs- oder katalytischen Prozessen, in dem ein oder mehrere Teilnehmer zusätzlich autokatalytisch wirken." [Jantsch, 1979]*

Als Beispiel führt Jantsch die Belousov-Zhabotinsky-Reaktion an, bei der Malonsäure durch Bromat in einer Schwefelsäurelösung in Gegenwart von Ceriumionen oxidiert. Bei bestimmten Mengenverhältnissen bilden sich konzentrische oder spiralförmige interferierende Wellen beobachten. Für

den informationstechnologisch interessierten Leser dürfte sich die schematische Darstellung als verständlicher und anregender erweisen. Abbildung 3 stellt diese Reaktion schematisch entsprechend [Jantsch, 1979] dar. A und B sind Anfangsprodukte, P und Q Endprodukte, X vemehrt sich autokatalytisch und zerfällt dabei in Q und bildet zugleich mit Y und Z einen crosskatalytischen Zyklus. Der Leser übersetze für ein theoretisches, autokatalytisches Software-System entsprechend Anfangsprodukte in Inputs, Ausgangsprodukte in Outputs und Katalyse in entsprechende Algorithmen.

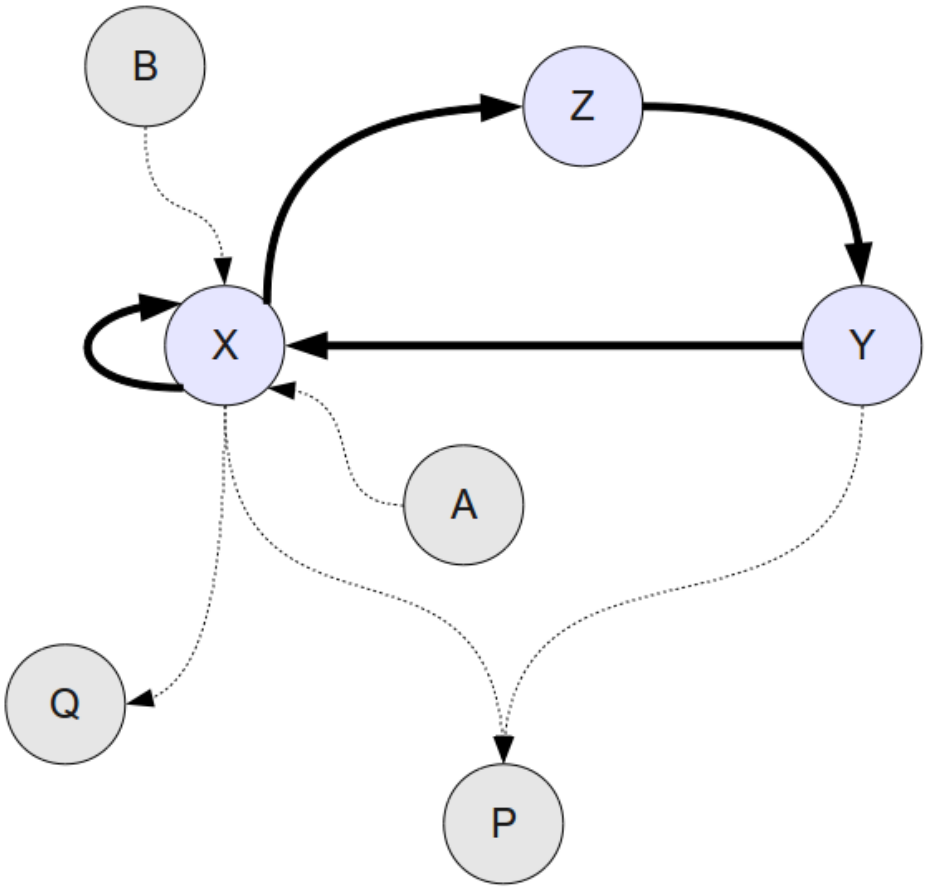

Abbildung 3: **Schematische Darstellung der Belousov-Zhabotinsky-Reaktion** Hyperzyklus hervorgehoben (nach [Jantsch, 1979])

5.2.2 Autopoiese

Maturana und Varela pägten den Begriff der Autopoiese:

Definition 13 *"Die Autopoietische Organisation wird als eine Einheit definiert durch ein Netzwerk der Produktion von Bestandteilen, die*
1. rekursiv an demselben Netzwerk der Produktion von Bestandteilen mitwirken, das auch diese Bestandteile produziert, und
2. das Netzwerk der Produktion als eine Einheit in dem Raum verwirklichen, in dem die Bestandteile sich befinden." [Maturana et al., 1974] nach [Mußmann, 1995]

Diese Definition gelte allgemein für lebende Systeme, die sich spontan durch Interaktion ansonsten unabhängiger Elemente selbstorganisiert als einheitliches Netzwerk ausbilden, wordurch sich zB. kognitive Systeme oder Zellen/Organismen entwickeln. [Mußmann, 1995]
Aufbauend auf den bisherigen Definitionen in dieser Arbeit, können wir vereinfacht zusammenfassen, dass ein System autopoietisch ist, wenn seine Komponenten sich selbst erneuern. Der Definition nach bilden X, Y und Z in der in Abbildung 3 schematisch dargestellten, chemischen Reaktion ein autopoietisches System, denn X erzeugt sich autokatalytisch selbst und X, Y und Z erzeugen sich ebenfalls selbst. Das simpelste autopoietische System ist also eines, das aus einer einzigen Komponente besteht, die sich autokataytisch selbst erneuert. Wie etwa X in unserem Beispiel.
Aus der Definition der Autopoiese folgt direkt die *operationale Geschlossenheit* und indirekt die *energetische Offenheit* (dadurch, dass die Komponenten durch irgendetwas erneuert werden müssen).

Die Entwicklung eines dissipativ selbstorganisierenden Systems ist eine *Evolution*. Im Gegensatz zur Devolution der konventional selbstorganisierenden Systeme arbeitet die Entwicklung nicht auf ein Equilibrium hinaus, sondern bleibt fern vom Gleichgewicht in einem andauernden inneren Zustand des Ungleichgewichts. [Jantsch, 1979] Daraus folgt, dass beobachtbare Ordnung des Systems - im Sinne von Selbstorganisation - in der Organisation des Systems liegt, die durchaus auch dynamisch sein kann. Eine Zelle bleibt eine Zelle, auch wenn sich nach und nach sämtliche Moleküle in ihr ersetzen und nur weil einige Zellen in unserem Gehirn gerade mit den katalysierten Bestandteilen unseres Frühstücks ausgetauscht werden, bedeutet das nicht, dass wir eine andere Person sind.

5.3 Feedback

In dieser Arbeit wird der Begriff *Feedback* (eingedeutscht aus dem Englischen für "Rückkopplung") der folgenden Definition entsprechend verstanden:

Definition 14 *"the return to the input of a part of the output of a machine, system, or process" [Merriam-Webster-Online, 2009]*

Diese Definition verlangt von der Struktur des Systems Zyklen zu enthalten, so dass ein Signal erneut auf die gleiche Weise modifiziert werden kann. Ein eingängiges Beispiel hierfür ist die Rückkopplung eines Mikrophons, das an den Lautsprecher gehalten wird, der die Aufnahme eben dieses Mikrophons ausgibt. Dieses Verstärken bezeichnet man als *positives Feedback* (Auch wenn es im Fall des Lautsprechers eher wenig positiv scheint). *Negatives Feedback* bezeichnet analog eine Signalabschwächung. Ein weiteres beliebtes Beispiel für Feedback ist das Sierpinski-Dreieck. Der Beginn sei ein schwarzes Dreieck, der Algorithmus laute: "Teile jedes Dreieck in vier gleichgroße Dreiecke, entferne das mittlere und beginne mit dem Output dieser Operation als Input von vorn". Abbildung 4 illustriert die Entwicklung.

Abbildung 4: **Entwicklung eines Sierpinski-Dreiecks**

Ein interessantes Phänomen, das durch Feedback enstehen kann, ist Skaleninvarianz (d.i. Gleichbleiben von Charakteristika des Beobachteten Objekts bei Skalierung). Wir betrachteten das Sierpinski-Dreieck bereits. Man kann es aufgrund seiner rekursiven Definition unendlichfach vergrößern, ohne, dass sich die Struktur verändert. Abbildung 5 zeigt drei Aufnahmen des Fraktalzoomprogramms xaos, die Skaleninvarianz unmittelbar verständlich machen. Die weiße Umrandung zeigt die Position der überlappenden, vergrößerten Aufnahme an.

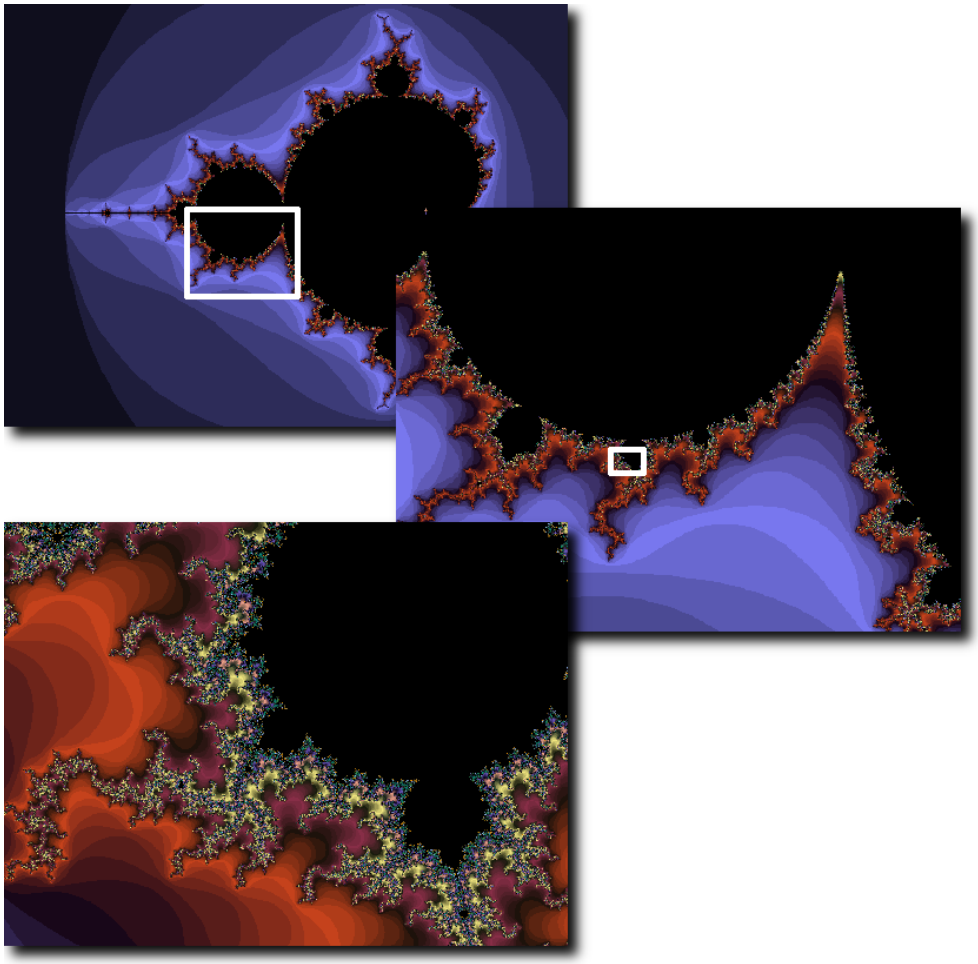

Abbildung 5: **Fraktalzoom in xaos**

5.4 Perturbation

Wir betrachten ein beliebiges P2P(Peer-To-Peer)-Filesharing-System. Auf mehreren, über das Internet miteinander verbundenen Computern, laufe die gleiche Software. Diese Software bilde mit ihren Installationen auf den anderen Computern ein Netzwerk. Die Operationsdetails werden hier vernachlässigt (siehe dafür exemplarisch den Abschnitt der Fallstudie zu Bittorrent). Weiter seien die Rechner absolut absturzsicher und das Netzwerk absolut zuverlässig um uns vom Thema abweichende Überlegungen zu ersparen. Wir betrachten dieses Netzwerk als ein System. Als Grenze dieses Systems sehen wir anwendungsorientiert die Schnittstelle zum Benutzer - eine hübsche, grafische Benutzeroberfläche, mittels derer er Dateien hoch- oder herunterladen kann.
Wir können das Netzwerk durch hinzufügen von Dateien oder neuer Instanzen der Software offensichtlich verändern. Für ein besseres Verständnis ändern wir unseren Standpunkt auf den des Systems. So sprechen wir davon, dass das System *Perturbationen* erfährt. Der Begriff der Perturbation ist eine Wortübernahme aus dem Spanischen. Der Übersetzer von [Maturana and Varela, 1987] erklärt:

> "Der von den Autoren verwendete Begriff *perturbación* bezeichnet (anders als *disturbación*, was eher negativ konnotiert ist) Zustandsveränderungen in der Struktur eines Systems, die von Zuständen in dessen Umfeld *ausgelöst* werden. ... wird in Abstimmung mit F. Varela im Folgenden von «Perturbation» und in der Verbform von «perturbieren» gesprochen"

[Maturana and Varela, 1987]
Auf solche Perturbationen reagiert dann das System; im Fall des Hochladens einer Datei etwa damit, dass es die Datei an die anderen Peers verteilt und wenn eine neue Programminstanz sich dem Netzwerk anschließen möchte, durch entsprechende Aufnahme in das Netzwerk.

5.5 Stigmergie

Operational offene Systeme können *Stigmergie* aufweisen. Stigmergie ist eine Form der Kommunikation, bei der Komponenten des Systems die Umgebung als Medium verwenden um mit anderen Komponenten indirekt zu kommunizieren. [Meer and Koppen, 2005]. Ein Beispiel, dessen

Modellierung in der IT Anklang gefunden hat, weil es akzeptable Lösungen für das Problem des kürzesten Weges (s.a. *Traveling Salesman*) findet, ist das der Ameisenkolonie. Abbildung 6 illustriert das Verfahren.

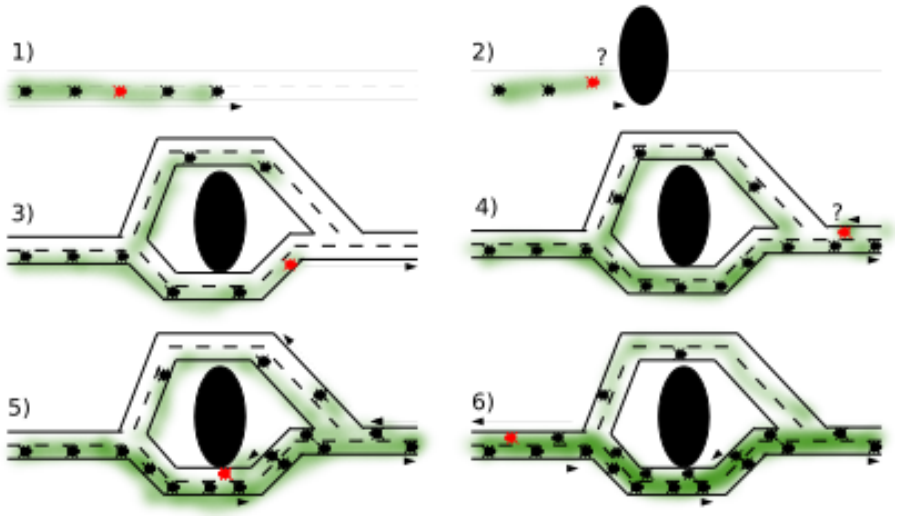

Abbildung 6: **Finden des kürzesten Weges durch Stigmergie** [Meer and Koppen, 2005], modifiziert durch den Autor

Ameisen hinterlassen auf Schritt und Tritt Pheromone: spezielle Duftstoffe, die andere Ameisen anlocken und die mit der Zeit dispergieren. In der Abbildung sind diese Pheromone als grüne Wolken dargestellt. Je höher die Intensität, desto kräftiger die Farbe.
Die Ameisen machen sich also auf den Weg Ressourcen zu sammeln; sie wissen nicht wo sich diese Ressourcen befinden, also laufen sie blind los. Die erste zumindest, die nachfolgenden Ameisen folgen der ersten aufgrund der von ihr hinterlassenen Pheromone (zumindest mit höherer Wahrscheinlichkeit, als ihr nicht zu folgen) (1). Nun stoesst die erste Ameise auf ein Hindernis (2) und entscheidet sich zufällig für eine der beiden Richtungen. Die zweite Ameise entsprechend folgt der ersten weiter mit höherer Wahrscheinlichkeit als dass sie ihr nicht folgt, aber es werden dank des Zufallselements immer Ameisen ausbrechen und neue Wege gehen (3). Auf dem Rückweg entscheiden sich die Ameisen erneut, welchem Weg sie folgen (4). Dadurch, dass die Ameisen auf dem unteren, kürzeren Weg schneller wieder zurück sind, wächst dort die Pheromonkonzentration schneller als oben, wodurch mehr Ameisen diesen Weg wählen werden (5). Bis auf wenige Ausnahmen werden bald sämtliche Ameisen dem kurzen Weg folgen (6). Dem Leser wird das zugrundeliegende Prinzip des positiven Feedbacks auffallen.

5.6 Kritikalität und die Edge of Chaos

Der Begriff der Kritikalität erlangte vor allem in der Thermodynamik große Berühmtheit, wo er die Grenze zweier Aggregatszustände bezeichnet. In diesem Zustand kann jede Komponente des Systems jede andere Komponente beeinflussen; eine Kettenreaktion ist möglich. [Meer and Koppen, 2005] Dies wird in der folgenden Definition zusammengefasst.

Definition 15 *Kritikalität bezeichne einen Zustand, in dem sich ein System befindet, wenn eine lokale Perturbation sich auf das gesamte System auswirken kann.*

Es Mangelt nicht an Veranschaulichungen der Kritikalität. Die Phrase "Der Tropfen, der das Fass zum Überlaufen bringt", beschreibt genau dieses Phänomen. Das Fass ist bis zum Rand mit Wasser gefüllt. Gerade so weit, dass es nicht überläuft. Perturbiert man nun auf beliebige Weise eine beliebige Stelle der Wasseroberfläche, setzt sich diese Perturbation in alle Richtungen fort, bis sie am Rand zum Überlaufen des Fasses führt. Die Wasseroberfläche befand sich in der Kritikalität.

In Forschungen mit zellulären Automaten prägte Christopher Langton den Begriff *Edge of Chaos* für einen Operationsbereich, in dem die Automaten die Grundlagen von Berechnungen unterstützen. Als diese Grundlagen zählt er auf: Informationsübermittlung, -speicherung und -modifikation. Er kam zu folgender Schlussfolgerung [Langton, 1990] (der Länge wegen vom Autor aus dem Englischen übersetzt):

> "Von Neumann beobachtete:
>
> > 'Da gibt es nun diese absolut maßgebende Eigenschaft von Komplexität, dass eine kritische Größe existiert, unterhalb derer ein Syntheseprozess degenerativ ist, aber oberhalb derer das Phänomen der Synthese explodieren kann, wenn entsprechend arrangiert. In anderen Worten: Wo Synthese von Automaten in solcher Weise voranschreiten kann, als dass jeder Automat andere Automaten produziert, die komplexer sind und mehr Potential haben, als er selbst.'
>
> Obwohl wir "Komplexität" in einem etwas anderen Sinn verwenden, als von Neumann, unterstützen die Ergebnisse dieses

> Papers seine Beobachtung. Wichtiger jedoch ist, dass wir ähnliche Beboachtungen für den Fall gemacht haben, dass zu viel "Komplexität" existiert: Ab einer gewissen Menge an "Komplexität" ist der Prozess der Synthese ebenso degenerativ.
> In anderen Worten: wir fanden heraus, dass Systeme eine obere und eine untere "Komplexitäts"-Grenze aufweisen, wenn der Prozess der Synthese nicht degenerativ, sondern konstruktiv und unbegrenzt sein soll. Wir fanden außerdem heraus, dass diese Grenzen relativ nah beieinander zu liegen scheinen und nahe einem Phasenwechsel auftreten.
> Da die Systeme nahe dem Phasenwechsel eine Reihe von Verhalten aufweisen, die die Phänomenologie von Berechnungen überraschend gut reflektieren, postulieren wir, dass wir Berechnungen innerhalb des Spektrums dynamischen Verhaltens an Phasenwechseln finden können - an der *"egde of chaos"*."

Diese Beobachtung ist von bemerkenswerter Bedeutung für die Software-Entwicklung. Da Computer offensichtlich kleiner und günstiger werden, wächst der Grad der realisierbaren Vernetzung proportional. Mit dem Grad der Vernetzung steigt der Grad der Komplexität der Kommunikation innerhalb des Netzwerkes. Im Speziellen ist es also für P2P-Anwendungen wichtig, dass sie nicht nur eine Mindestkommunikationskomplexität aufweisen, sondern unterhalb einer gewissen maximalen Komplexität liegen müssen, um produktiv zu bleiben. Voraussetzung ist, dass sich Langtons Ergebnisse auf P2P-Netzwerke übertragen lassen. Es scheint kein Hindernis dafür zu geben und nachfolgend skizzierte naive Überlegungen verdeutlichen die Übertragbarkeit.
Man denke an ein P2P-Filesharing-Programm mit Suchfunktion; das Netzwerk sei ein *Darknet*, dh. jeder Knoten sei nur mit vertrauten Nachbarn verbunden. Die Suchfunktion suche das gesamte Netzwerk ab. Eine solche Suchfunktion kann nur in einem gewissen Rahmen zufriedenstellend funktionieren: Zu wenig Peers bedeuten zu wenig Information, zu viele Peers bedeuten zu tiefe Suchbäume.

Die Universalität von Kritikalität wird offensichtlich, wenn wir Kritikalität in der Gruppendynamik suchen.

> "Zunächst ist zu klären, wie groß eine Gruppe überhaupt sein kann. Die Gruppengröße ist dabei abhängig von der anzustrebenden Kommunikationsdichte. Wenn man den Leistungsvor-

> teil von Gruppen (gegenüber einer bloß hierarchischen Steuerung) generieren will, dann müssen alle Mitglieder einer Gruppe sich an den Kommunikationsprozessen beteiligen (können). Die Fähigkeit dazu endet etwa bei 10-15 Gruppenmitgliedern. Für Arbeitsgruppen gilt die Zahl 5 bis 7 als optimal, in Führungssituationen gilt die Führungsspanne von 1 zu 10 als nicht zu überschreiten (1 Chef kann kommunikativ maximal 10 Mitarbeiter führen). Wächst eine Gruppe über diese kritische Größe hinaus, dann ist es zweckmäßig, die Gruppe zu teilen.
>
> Mit Sicherheit ist auszuschließen, dass eine Organisation kommunikativ wie eine Gruppe funktioniert. Organisationen sind strukturell gesehen „Gruppen von Gruppen“, die über Schaltstellen, Zwischenvorgesetzte, Sprecher etc. miteinander verknüpft sind. In Organisationen herrscht das Prinzip der „indirekten Kommunikation“, in Gruppen dagegen wird „face-to-face“ kommuniziert.” [Krainz, 2006]

Die Übergänge sind hier weicher, was uU. auch an der Natur der qualitativen Betrachtungsparadigma der Gruppendynamik liegt. Auffällt aber dennoch, dass eine Gruppe mit sehr wenigen Mitgliedern nicht funktioniert. Man würde zwei Leute nicht als Gruppe bezeichnen wollen. Fünf bis sieben Mitglieder ist der Optimale Bereich und danach zerfällt die Gruppe mit wachsender Zahl der Mitglieder in organisatorisches Chaos. In Arbeitsgruppen darf dies nicht passieren, deshalb findet hier ein “Phasenwechsel” statt und die Gruppe geht in eine Organisation (im soziologischem Sinn) über.

6 Fallbeispiel: Bittorrent

Bittorrent ist ein von Bram Cohen seit 2001 entwickeltes P2P-File-Distribution-Protokoll. Vorgestellt wurde es zuerst auf dem CodeCon 2002 [Cohen, 2002]. Im Jahr 2003 bestand das Frontend des Clients aus einem Dialogfenster, das aus dem Browser heraus gestartet wurde und minimale Informationen anzeigte. [Cohen, 2003] Das Protokoll verbreitete sich seither weltweit und ist inzwischen sehr populär. Für 2004 wurde berichtet, dass Bittorrent 50% des Gesamtdatenverkehrs im Internet ausmachte. [Dale et al., 2008] Die folgende systemische Analyse beginnt mit der Beschreibung des Bittorrentprotokolls und untersucht darauf aufbauend das Sys-

tem, welches durch Implementierung des Protokolls entsteht, auf seine Eigenschaften, insbesondere als selbstorganisierendes System.
Da ein System eine Einheit ist, stellt sich die Frage, wodurch diese Einheit gegeben wird. Diese Arbeit unterscheidet zwischen zwei Systemen. Das erste System, das wir betrachten wollen, ist das Netzwerk, das duch die Bittorrent-Client-Programme untereinander aufgespannt wird, auch *"Schwarm"* genannt [Dale et al., 2008]. Die zweite Sicht umfasst das komplette, durch das Protokoll erzeugte System, das einige Komponenten mehr enthält.

6.1 Protokoll

Die folgende Protokollbeschreibung basiert auf der offiziellen Bittorrent-Spezifikation [Cohen, 2008], sowie der Anwendungsbeschreibung des frühen Originalclients. [Cohen, 2003]

6.1.1 Anwendung

Die typische Anwendung von Bittorrent gliedert sich in zwei Abläufe: Je einen für den Besitzer der Datei, die er verteilen möchte und einen für die Interessenten an dieser Datei. Wir nennen den Besitzer *Original-Seeder*, die Datei entsprechend *Original*. Die am Original interessierten Benutzer nennen wir *Leecher*. Als *Peer* bezeichnen wir je nach Kontext einen Benutzer des Netzwerks oder die Instanz seines Bittorrent-Clients. Jeder Peer, der die Datei vollständig heruntergeladen hat und weiterhin Teil des Systems ist, gilt als *Seeder*.
Original-Seeder:

1. Um den Lebenszyklus einer Datei-Distribution mit dem Bittorrent-Protokoll zu beginnen, benötigt der Original-Seeder eine mediative Serverinstanz, die *Tracker* genannt wird und die in der Vanilla-Version (dh. ohne Erweiterungen) des Protokolls die zentrale Anlaufstelle für alle Peers darstellt.

2. Nun erstellt der Original-Seeder eine Datei (*.torrent Datei* genannt), die Metainformationen zum Original und dessen Verteilung enthält. Vorallem die URL des Trackers, damit die Clients der Leecher sich automatisch zum korrekten Tracker verbinden.

3. Die .torrent Datei stellt er dem Publikum, an das er das Original senden möchte zur Verfügung. In der Regel geschieht dies über eine Webseite im WWW.

4. Jetzt startet der Original-Seeder einen Bittorrent-Client, lädt die .torrent Datei und wartet auf Leecher, die diese Datei herunterladen möchten.

5. Der Client verbindet sich zum Tracker und registriert dort, dass er diese Datei besitzt.

Leecher:

1. Die Interessenten laden die .torrent Datei herunter und öffnen sie mit einem Bittorrent-Client.

2. Der Client bekommt vom Tracker die Verbindungsinformationen der bereits am Tracker angemeldeten Peers und beginnt den Download.

6.1.2 Bencoding

Im Bittorrent-Protokoll werden Informationen häufig *benkodiert* dargestellt. Dies ist eine Syntax zur kodierung von Informationen, die diese Elemente kennt:

- *Dictionary*: Eine Sequenz von Schlüssel-Wert Paaren, wobei die Schlüssel Strings (Im Sinne des Elements *String*, nicht einer beliebigen Zeichenkette) sind und die Werte beliebige Elemente dieser Notation.
- *List*: Eine Sequenz von Elementen
- *String*: Eine UTF8 Zeichenkette mit vorausgehender Längenangabe
- *Integer*: Eine ganze Zahl, wobei Null nicht negativ sein darf

Bencoding entspricht der folgenden EBNF:

```
(* Basisdefinitionen, UTF8 nicht ausspezifiziert *)
utf8 = <any valid string of UTF-8 characters>
zero = "0";
nonzero = "1" | "2" | "3" | "4" | "5" |
```

```
          "6" | "7" | "8" | "9";
number = nonzero , { nonzero | "0" };

(* Liste gültiger Elemente *)
element = dictionary | list | string | integer;

(* Elementspezifikationen *)
integer = "i" , (["-"] , number) | zero , "e";
string = number , ":" , utf8;
list = "l" , { element } , "e";
dictionary = "d" , { string , element }, "e";
```

6.1.3 .torrent Dateien

Diese Dateien sind benkodierte Dictionaries und sind wie nachfolgend aufgebaut. Für detailliertere Spezifikationen, sowie zahlreiche offizielle und inoffizielle Erweiterungen ist dem Leser zusätzlich zu [Cohen, 2008] eine Internetrecherche empfohlen. Um zu verdeutlichen, dass die hier aufgeführten Schlüssel nicht etwa als Zeichenketten (zB. “announce”), sondern vielmehr als ihr benkodiertes Äquivalent (zB. “8:announce”) vorliegen, wurden sämtliche Elemente des Bencodings *kursiv* gesetzt, statt in Anführungszeichen.

- *announce* - die URL des Trackers
- *info* - ein neues *Dictionary*, das sich unterscheidet, je nachdem ob das Original aus einer oder mehrerer Dateien besteht:
  - *name* - eine Empfehlung für den Dateinamen des Downloads (oder den Namen des Verzeichnisses im Fall mehrerer Dateien)
  - *piece length* - die Größe sogenannter *Chunks* als Potenz von 2 in Kilobytes - Ein Chunk ist ein Datenblock in welche das Original zerlegt wird, um es leichter zu verteilen. Die Regelgröße beläuft sich auf 512KB, was einem Wert von 9 entspricht.
  - *pieces* - ein *String* der Länge Dateigröße / piece length * 20 - Dieser *String* ist eine Konkatenation von SHA1-Prüfsummen der entsprechenden Chunks, welche der Überprüfung einer korrekten Dateiübertragung dienen.
  - *length* (falls nur eine Datei) - die Größe der Datei in Bytes

- *files* (falls mehrere Dateien) - eine *List*, die aus *Dictionaries* mit den folgenden Schlüssel-Wert Paaren besteht:
  * *length* - die Größe dieser Datei in Bytes
  * *path* - eine *List* von einem oder mehr *Strings* die den Pfad der Datei repräsentieren - "pfad/zu/datei.txt" wird als "l4:pfad2:zu9:datei.txte" kodiert. So wird die Verzeichnisstruktur abgebildet, ohne betriebssystemabhängige Konventionen mit zu kodieren, wie etwa das Trennzeichen zwischen Ordner und Unterordner, wodurch Portabilität erreicht wird.

6.1.4 Tracker-Kommunikation

Der Tracker wird in von ihm bestimmten Zeitabständen (s.u.) über das HTTP-Protokoll mit GET-Befehl abgrufen, mit den folgenden Parametern (HTTP-entsprechend escaped) gesetzt:

- "info_hash" - ein 20 Byte SHA1 Hash des benkodierten *info* Feldes der .torrent Datei.
- "peer_id" - eine 20 Zeichen lange Zeichenkette, die die ID dieses Peers darstellt - Diese IDs werden vom Client selbst generiert (als Zufallszahl), beim Start eines neuen Downloads.
- "ip" (optional) - die IP oder der Hostname des Peers
- "port" - die Portnummer auf der der Client horcht (Default ist 6881)
- "uploaded" - die bisher hochgeladene Datenmenge (es wurde keine Einheit festgeschrieben)
- "downloaded" - die bisher hochgeladene Datenmenge (es wurde keine Einheit festgeschrieben)
- "left" - die Menge an Bytes, die dieser Downloader noch herunterladen muss.
- "event" (optional) - Einer der Werte:
  - "started" - Der Download wurde gerade gestartet.
  - "completed" - Der Download wurde erfolgreich abgeschlossen.
  - "stopped" - Der Download wurde unterbrochen.
  - "empty" (oder weggelassen) - Der Download läuft und dies ist eines der regelmäßigen Updates.

Die Antwort des Trackers ist ein benkodiertes *Dictionary* der Form:

- *failure reason* (falls ein Fehler aufgetreten ist) - ein *String*, der eine menschenlesbare Fehlermeldung enthält
- *interval* - Zeit in Sekunden, die der Downloader zwischen Tracker-Anfragen warten soll
- *peers* - eine *List* von *Dictionaries*, die die folgenden Peer-Informationen enthalten:
  - *peer id* - die ID des Peers
  - *ip* - die IP des Peers
  - *port* - der Port, auf dem der Peer horcht

6.1.5 Peer-Kommunikation

Mit den vom Tracker erhaltenen Informationen verbindet sich der Client mit den Peers, die ihm übermittelt wurden. Dies geschieht über eine symmetrische TCP-Verbindung. Die erste Nachricht, die von beiden Peers übermittelt wird, bildet einen Handshake. Danach werden Nachrichten ausgetauscht, bis die Verbindung von einem der Peers getrennt wird. Der Handshake besteht aus je einer Nachricht von Peer A an Peer B und umgekehrt und hat folgendes Format:

- $1Byte$ - Länge des folgenden Versions-Strings
- $NByte$ - Versions-String, N ist die vorausgegangene Länge
- $8Byte$ - reserviert für mögliche Protokollerweiterungen, sonst alle 0
- $20Byte$ - SHA1 Hash des benkodierten *info* Feldes aus der .torrent Datei. Dies ist der gleiche Hash, wie er auch als "info_hash" an den Tracker geschickt wurde, mit dem Unterschied, dass er hier nicht für HTTP escaped wird.
- $20Byte$ - *peer id*, wie sie an den Tracker geschickt wird und wie sie in den Peer-Listen in der Antwort des Trackers vorkommt.

Nachdem diese Formalien abgeschlossen sind, machen sich die Peers an den Informationsaustausch. Hier führen wir sog. *Choking* ein. Choking, zu deutsch etwa "Zudrehen" im Sinne von "Die Leitung zudrehen", bedeutet, dem Gegenüber keine Daten zu schicken. Ob der Peer bei seinem

Gegenüber *choked* ist, muss vom Client behalten werden, damit er weiss, dass er keine Anfragen stellen braucht, da diese ohnehin verworfen würden. Ein weiteres Flag muss erinnert werden: das *interested* flag, das besagt, ob das Gegenüber an irgendwelchen Chunks interessiert ist, die wir bereits heruntergeladen haben. Nach dem Handshake stehen beide Clients auf *choked* und *uninterested*.

In den folgenden Nachrichten bestehen alle Zahlen aus 4 Byte langen big-endian kodierten Integers, nachfolgend mit *int* bezeichnet.
Nachrichten beginnen jeweils mit einem int, das ihre Länge angibt, gefolgt vom Rumpf der Nachricht, der aus einem Typ-Byte und eventuellem Payload besteht. Eine Längenangabe von 0 wird als Keep-Alive-Paket verwendet und hat weder Typ noch Payload. Die Verschiedenen Nachrichtentypen und ihre Funktionen lauten wie folgt: (Der Eingängigkeit wegen in der ersten Person, aus der Sicht des Clients formuliert.)

- 0 (*choke*) - *"Ich drehe Dir die Leitung zu."* (Kein Payload)
- 1 (*unchoke*) - *"Ich drehe Dir die Leitung auf."* (Kein Payload)
- 2 (*interested*) - *"Ich bin an Chunks interessiert, die Du anbietest."* (Kein Payload)
- 3 (*not interested*) - *"Ich bin derweil an nichts interessiert, was Du mir anbietest."* (Kein Payload)
- 4 (*have*) - *"Ich habe diesen Chunk vollständig heruntergeladen."* Diese Nachricht wird immer an alle Peers geschickt, wenn der Client ein Chunk heruntergeladen hat. Der Payload besteht aus einem int, das den Index des Chunks abbildet (von 0 weg durchnummeriert).
- 5 (*bitfield*) - *"Schau, was ich tolles habe!"* Diese Nachricht wird direkt nach dem Handshake geschickt, sofern der Client bereits Chunks vollständig heruntergeladen hat, damit das Gegenüber entscheiden kann, ob es daran interessiert ist, oder nicht. Der Payload besteht aus einer Bitmaske, bei der jedes Bit für einen Chunk steht. Ein gesetztes Bit bedeutet, dass der Chunk vollständig vorliegt.
- 6 (*request*) - *"Ich hätte gern diese Daten von Dir."* - Bedeutet dem anderen Peer, welche Daten er senden soll. Der Payload besteht aus drei ints: dem Chunk-Index, einem Byte-Offset innerhalb des Chunks und der Anzahl von Bytes, die Übertragen werden sollen.

Außer, wenn das Chunk-Ende erreicht wird, sollte die Anzahl eine Zweierpotenz sein.

- 7 (*piece*) - *"Hier sind die Daten, die Du wolltest."* Besteht aus zwei ints und einem Datenblock variabler Länge. Der erste int ist der Chunk-Index, der zweite der Byte-Offset innerhalb des Chunks. Danach folgen die Nutzdaten.
- 8 (*cancel*) - *"Ich will diese Daten doch nicht."* Die Signatur und der Inhalt ist deckungsgleich mit dem der korrospondierenden, vorrausgegangenen Typ 6 (*request*) Nachricht.

6.2 Untersuchung des Schwarms

Der Schwarm besteht aus den Peers im Sinne der Clients, die Daten miteinander austauschen können, jenachdem, wie sie sich mit *interested* und *choked* einigen. In dieser Systembetrachtung beobachten wir das Netzwerk der Datenübertragung. Der Veranschaulichung wegen fließen Daten in den folgenden beiden Abbildungen hierarchisch von oben nach unten durch das Netzwerk. Das bedeutet der jeweils obere hat die unteren Peers, mit denen er verbunden ist, *unchoked* und diese sind im Gegenzug *interested* am oberen Peer. Abbildung 7 illustriert die Entwicklung mit Farbmischung. Im Zustand 1) sehen wir die beiden Leecher-Clients, wie sie eine Datenverbindung aufgebaut haben. 2) zeigt die beiden Leecher-Clients mit unterschiedlichen Chunks des Originals. 3) zeigt, dass die Leecher-Clients nun untereinander die fehlenden Chunks austauschen.

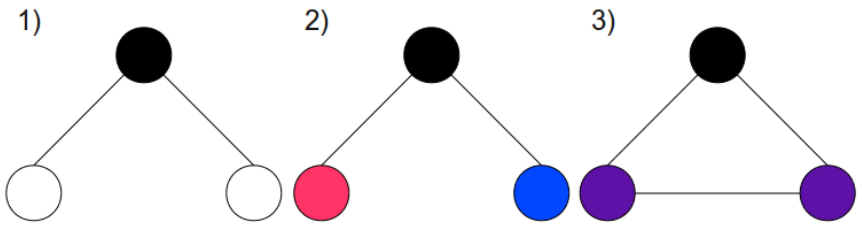

Abbildung 7: **Bittorrent-Schwarm** - Entwicklung mit einem Seeder und zwei Leechern

Abbildung 8 zeigt dieses Beispiel erweitert um einige Peers. Wir können einen neuen Seeder beobachten, der überhaupt keinen Kontakt zum Original hatte, sondern durch *Rot*, *Grün* und *Blau* die gewünschten Daten komplett herunterladen konnte.

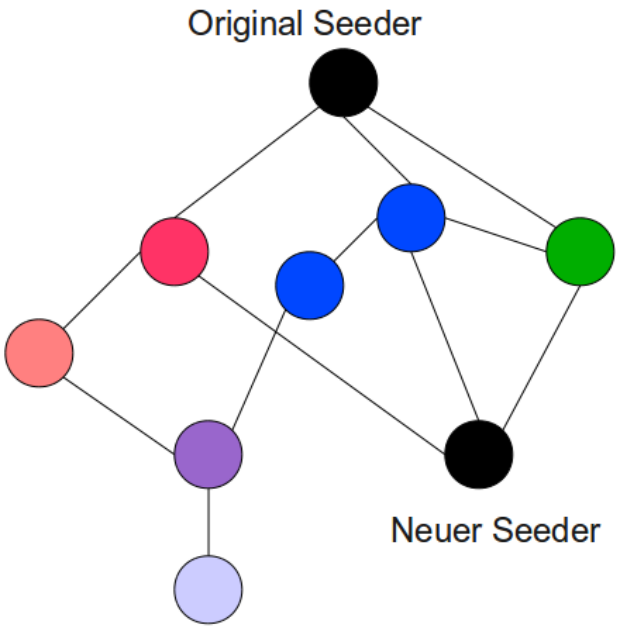

Abbildung 8: **Bittorrent-Schwarm** - Entwicklung mit komplexerem Netzwerk

Wir beobachten ein konventionell selbstorganisierendes System, das dem strukturellen Gleichgewicht der vollständigen Verteilung des Originals entgegenstrebt. Es ist ein emergentes Phänomen beobachtbar: Angenommen der Original-Seeder in Abbildung 8 kappt seine Verbindung zum Netzwerk, bevor der neue Seeder existiert. Zu diesem Zeitpunkt, kann es sein, dass das System vollständiges Wissen über das Original hat, das keine der Komponenten mehr aufweist. In der Praxis bedeutet das, dass auch Downloads abgeschlossen werden können, wenn keine Seeder mehr im Netzwerk vorhanden sind.
Ausgehend von diesen Beobachtungen, kann man Netzwerke aus Momentaufnahmen des Schwarms aufbauen, die den verschiedenen Zuständen der Peers entsprechen:

- Verbindungsnetzwerk - das Netzwerk aller miteinander Verbundenen Peers
- *(Un)interested*-Netzwerk - das Netzwerk der Peers, die an anderen interessiert sind
- *(Un)choked*-Netzwerk - das Netzwerk, das durch choken von Peers entsteht

- Datenübertragungsnetzwerk - Netzwerk der Downloads - Abbildungen 7 und 8 gehören zu dieser Art von Netzwerk.

[Dale et al., 2008] führten Netzwerkanalyen eben dieser (mit Ausnahme von *uninterested* und *choked*) Netzwerke durch und fanden dabei heraus, dass das Unchoked-Netzwerk ein *Scale-Free-Network* (Eine Konfiguration, in der es viele Knoten mit wenigen Verbindungen und wenig Knoten mit vielen Verbindungen gibt.) ist, was es besonders robust gegen bestimmte Angriffe macht.

6.3 Untersuchung des vollständigen Systems

Die vorausgegangenen Untersuchungen bezogen sich nur auf einen kleinen Teil des kompletten Bittorrent-Protokolls. Cohen spezifiziert, wie folgt:

> "A BitTorrent file distribution consists of these entities:
>
> - An ordinary web server
> - A static 'metainfo' file
> - A BitTorrent tracker
> - An 'original' downloader
> - The end user web browsers
> - The end user downloaders
>
> " [Cohen, 2008]

Mit "downloader" ist hier der Client gemeint. Abbildung 9 zeigt das vollständige Bittorrent-System inklusive deren Endbenutzer. Die drei abgebildeten Peers entsprechen dabei denen aus Abb. 7, Tracker-Kommunikation ist blau dargestellt, Aktionen des Endbenutzers in dunkelrot. Der Webserver enthält die .torrent Datei.
Es handelt sich aufgrund des zugrundeliegenden Netzwerkes ebenfalls um ein konservativ selbstorganisierendes System. Dessen Grenze ist durchlässig für Steuerbefehle der Benutzer an die Clientprogramme, sowie Daten. Die Steuerbefehle an den Webserver wurden der Übersichtlichkeit halber nicht eingezeichnet. Den Richtungspfeilen kann man Wirkungspropagation entnehmen. So beeinflusst ein Leecher theoretisch die Verbindungen aller Peers des Netzwerkes. In der Praxis macht sich das zB. dadurch bemerkbar, dass, um das System funktionierend zu halten, das durchschnittliche Verhältnis von Upload zu Download ausgeglichen sein muss. (s.a. *"Free Rider Problematik"*)

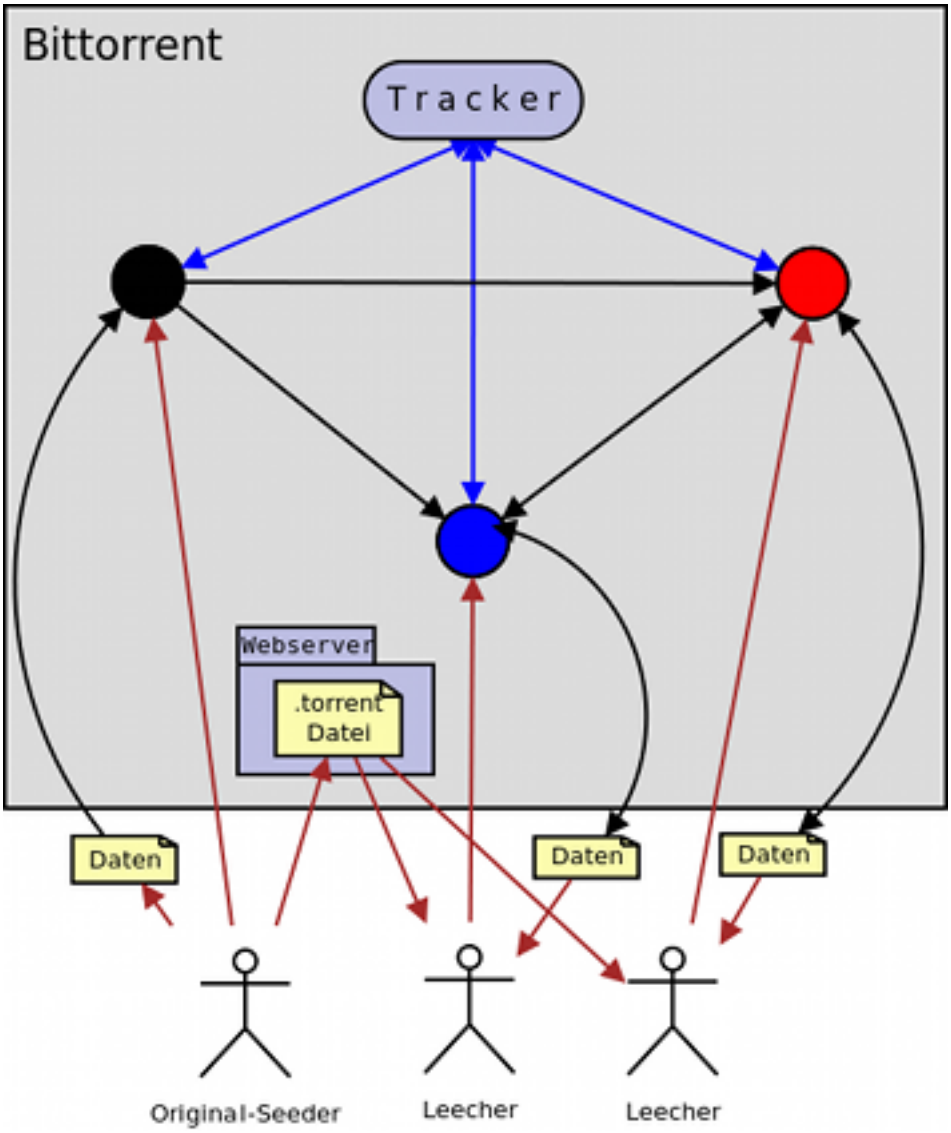

Abbildung 9: **Vollständiges Bittorrent-System**

Ebenfalls leicht erkennbar ist der Unterschied zwischen Organisation und Struktur des Systems. Wir können sämtliche Clients durch andere ersetzen, ebenso wie den Webserver und den Tracker, solange sie der Protokollspezifikation entsprechen, ohne das System zu verlieren, was uns eine hohe Robustheit gegenüber Ausfällen garantiert.
Würde man die Benutzer ebenfalls als Teil des Systems sehen, die (auto)-katalytisch neue Originale erstellen und über Bittorrent verteilen, so erhielte man ein autopoietisches System, das evolviert und wächst, solange es nicht durch gewisse Lobbies daran gehindert wird. Entsprechend der eingangs erwähnten autokatalytischen Reaktion, verwenden die Benutzer über Bittorrent bezogene Information um daraus eigene Information zu schaffen, die wiederum über Bittorrent verteilt wird. Als Beispiel denke man an Remixes musikalischer Werke. Durch diesen kreativen Prozess wird das System aufrecht erhalten. Die Systemgrenzen sind mehrdimensional: Zum einen bildet sich ein Overlay-Netzwerk aus Bittorrentkommunikation im unterliegenden Netz (idR. das Internet), zum anderen bilden die Benutzer des Protokolls eine Gruppe. Beide dieser Grenzen sind organisatorisch geschlossen: das Netzwerk wird durch das Bittorrent-Protokoll spezifiziert, die Benutzerkommunikation durch soziale Faktoren, sowie Kommunikationsmöglichkeiten im Internet. Dennoch sind beide strukturell offen, denn es können sowohl neue Clients, neue Downloads als auch neue Benutzer dem System beitreten und es wachsen lassen. Die Benutzer erhalten zudem sämtliche Bestandteile dieses Systems. Server und Software wird verbessert oder erneuert, wenn nötig.

7 Schluss

Das schöne an Systemdenken ist in den Augen des Autors seine Universalität. Die systemische Untersuchung des Bittorrent-Protokolls beispielsweise enthüllte das emergente Phänomen der Informationsverteilung. Dieses Wissen kann man nun versuchen auf andere, eventuell komplexere Gebiete übertragen. So könnte man daraus ableiten, dass die immer weitere Spezialisierung von Ausbildungen nicht etwa dazu führt, dass keine Allgemeinbildung mehr existiert, sondern sehr viel breiteres Wissen zur Verfügung steht, solange dieses Wissen kommuniziert wird.
Durch Erkennen von Autopoiese im Bittorrent-System inklusive Benutzern, kann man die Frage aufwerfen, ob dieses System lebendig ist und eine Art Meta-Menschen formt. Weiter kann gefragt werden, ob ein beliebiges System chaotisch (lebendig, selbstorganisierend) wird, wenn man

eine (oder mehrere) chaotische (lebendige, selbstorganisierende) Komponente(n) hinzufügt. Ergebnisse solcher Überlegungen könnten, wenn sie universell genug sind, wieder rückwirkend auf zB. Bittorrent angewendet werden und dadurch seine Systemeigenschaften gezielt verändern und wiederum neue Erkenntnisse aus diesen Veränderungen gezogen werden. Es ist diese Rekursion, die das System menschlicher Erkenntnis evolviert.

Literatur

[Ackoff, 1961] Ackoff, R. L. (1961). Systems, organizations, and interdisciplinary research. *General Systems (Yearbook of the Society for the Advancement of General Systems Theory)*, 5:1.

[Aristoteles,] Aristoteles. Metaphysik.

[Bar-Yam, 1997] Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Westview Press.

[Cohen, 2002] Cohen, B. (2002). Codecon 2002. `http://www.codecon.org/2002/program.html`.

[Cohen, 2003] Cohen, B. (2003). Incentives build robustness in bittorrent. `http://www.bittorrent.org/bittorrentecon.pdf`.

[Cohen, 2008] Cohen, B. (2008). The bittorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`.

[Dale et al., 2008] Dale, C., Liu, J., Peters, J., and Li;, B. (2008). Evolution and enhancement of bittorrent network topologies. In *16th International Workshop on Quality of Service*, pages 1–10.

[Hall and Fagen, 1956] Hall, A. and Fagen, R. (1956). Definition of system. *General Systems (Yearbook of the Society for the Advancement of General Systems Theory)*, 1:18.

[Hanh, 1999] Hanh, T. N. (1999). *The Heart of the Buddha's Teaching*. Broadway Books.

[Jantsch, 1979] Jantsch, E. (1979). *Die Selbstorganisation des Universums*. Carl Hanser Verlag.

[Kant, 1786] Kant, I. (1786). Metaphysische anfangsgründe der naturwissenschaft.

[Kornfield, 2001] Kornfield, J. (2001). *Seeking the Heart of Wisdom*, chapter 4, pages 38–56. Shambhala.

[Krainz, 2006] Krainz, E. E. (2006). *Gruppendynamik als Wissenschaft*, pages 7–28. Peter Heintel: betrifft: TEAM. Dynamische Prozesse in Gruppen. VS Verlag für Sozialwissenschaften.

[Langton, 1990] Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42:12–37.

[Leven et al., 1989] Leven, R. W., Koch, B.-P., and Pompe, B. (1989). *Chaos in dissipativen Systemen*. Vieweg.

[Maturana and Varela, 1987] Maturana, H. and Varela, F. (1987). *Der Baum der Erkenntnis*. Scherz Verlag.

[Maturana et al., 1974] Maturana, H., Varela, F. J., and Uribe, R. (1974). Autopoiese: die organisation lebender systeme, ihre nähere bestimmung und ein modell. In *Erkennen: Die Organisation und Verkörperung von Wirklichkeit. Ausgewählte Arbeiten zur biologischen Epistemologie*, pages 157–169. Maturana.

[Meer and Koppen, 2005] Meer, H. D. and Koppen, C. (2005). *P2P Systems and Applications*, volume 1, chapter 15 Characterization of Self-Organization, pages 227–246. Springer-Verlag.

[Merriam-Webster-Online, 2009] Merriam-Webster-Online (2009). feedback. `http://www.merriam-webster.com/dictionary/feedback`.

[Mußmann, 1995] Mußmann, F. (1995). *Komplexe Natur, komplexe Wissenschaft*. Leske + Budrich.

[Stein and Varela, 1993] Stein, W. D. and Varela, F. J. (1993). *Thinking About Biology*. Addison-Wesley.

[von Förster, 1998] von Förster, H. (1998). *Wahrheit ist die Erfindung eines Lügners*, chapter Management, page 92. Carl-Auer-Systeme Verlag.