Nunmehr zum achten Male liegt ein Sammelband zum Workshop „GeNeMe – Gemeinschaften in Neuen Medien“ vor, der Beiträge zu folgenden Themenfeldern enthält:

- Konzepte für GeNeMe (Geschäfts-, Betriebs- und Architektur-Modelle),
- IT-Unterstützung (Portale, Plattformen, Engines) von GeNeMe,
- E-Learning in GeNeMe,
- Wissensmanagement in GeNeMe,
- Anwendungen und Praxisbeispiele von GeNeMe und
- Soziologische, psychologische, personalwirtschaftliche, didaktische und rechtliche Aspekte von GeNeMe.

Sie wurden aus einem breiten Angebot interessanter und qualitativ hochwertiger Beiträge zu dieser Tagung ausgewählt.

Das Interesse am Thema GeNeMe (Virtuelle Unternehmen, Virtuelle Gemeinschaften etc.) und das Diskussionsangebot von Ergebnissen zu diesem Thema sind im Lichte dieser Tagung also ungebrochen und weiterhin sehr groß.

Die thematischen Schwerpunkte entsprechen aktuellen Arbeiten und Fragestellungen in der Forschung wie auch der Praxis. Dabei ist die explizite Diskussion von Geschäfts- und Betreibermodellen für GeNeMe, insbesondere bei der aktuellen gesamtwirtschaftlichen Lage, zeitgemäß und essentiell für ein Bestehen der Konzepte und Anwendungen für und in GeNeMe.

In zunehmendem Maße rücken weiterhin auch Fragen nach den Erfolgsfaktoren und deren Wechselbeziehungen zu soziologischen, psychologischen, personalwirtschaftlichen, didaktischen und rechtlichen Aspekten in den Mittelpunkt. Deshalb wurde hierzu ein entsprechender Schwerpunkt in der Tagung beibehalten.

Konzepte und Anwendungen für GeNeMe bilden entsprechend der Intention der Tagung auch weiterhin den traditionellen Kern und werden dem Anspruch auch in diesem Jahr gerecht.

Die Tagung richtet sich in gleichem Maße an Wissenschaftler wie auch Praktiker, die sich über den aktuellen Stand der Arbeiten auf dem Gebiet der GeNeMe informieren möchten.

www-mmt.inf.tu-dresden.de/geneme

ISBN: 3-86005-491-0

Meißner / Engelien • Virtuelle Organisation und Neue Medien 2005

TECHNISCHE UNIVERSITÄT DRESDEN

Klaus Meißner / Martin Engelien (Hrsg.)

Virtuelle Organisation und Neue Medien 2005

Workshop GeNeMe2005
Gemeinschaften in Neuen Medien

TU Dresden, 6./7.10.2005

TECHNISCHE UNIVERSITÄT DRESDEN

A.3 Rahmen für eine Governance in Open-Source-Projekten

Christoph Lattemann, Stefan Stieglitz
Universität Potsdam, Juniorprofessur für Corporate Governance & eCommerce

1. Einführung

Der Markt für lizenzgebundene Software ist durch starke wettbewerbliche Kräfte geprägt. Dies hat in den letzten Jahren zu einem Verdrängungswettbewerb geführt und resultiert heute in bestimmten Segmenten in oligopolistischen Strukturen. Umso erstaunlicher ist es, dass sich in diesem Umfeld erfolgreich und nachhaltig Open-Source-Projekte als Formen von wertschöpfenden Nonprofit-Organisationen bilden.

Die Motivation zur aktiven Teilnahme der Mitglieder in diesen Projekten entspringt nicht, wie bei kommerzieller Software-Produktion, der finanziellen Sphäre, vielmehr dominieren Aspekte der intrinsischen Motivation wie Spaß, Wissensdrang und die Möglichkeit zur Weiterbildung. Extrinsische Motive nicht monetärer Art wie Reputation, Identifikationsprozesse in Gruppen oder Karrieregedanken [Lerner & Tirole 01] stellen ebenfalls Grundvoraussetzungen für die erfolgreiche Durchführung von Open-Source-Projekten dar [Achtenhagen et al. 03].

Damit netzwerkartig strukturierte Organisationseinheiten, wie Open-Source-Projekte, erfolgreich operieren können, sind die organisatorischen Regelungen sowie Motivations- und Anreizsysteme so zu gestalten, dass die Organisationsmitglieder auch unter sich verändernden Rahmenbedingungen ihren Beitrag nicht nur einbringen können (Koordinationsprinzip) sondern dies auch aktiv und zum Teil selbstgesteuert wollen (Motivationsprozess). Governance wird im Rahmen dieser Untersuchung als institutionelle Steuerung zur Überwachung und Kontrolle verstanden [Schneider & Kenis 96]. Traditionelle agency-basierte Governance-Ansätze, die sich mit der Überwachung und Steuerung von unternehmensinternen Prozessen beschäftigen, vernachlässigen zum großen Teil „weiche“ Faktoren und rücken direkte Kontroll-, Anreiz- und Sanktionsmechanismen, die monetärer oder existenzieller Art sind, in den Vordergrund.

Um Governance-Mechanismen in Open-Source-Projekten zu untersuchen, muss von der streng an der Neuen Institutionenökonomie ausgerichteten Betrachtung und den bekannten Governance-Theorien abstrahiert werden. Psychologische und soziologische Aspekte müssen Berücksichtigung finden. In dem, in den 1990er Jahren entwickelten

Stewardship-Ansatz, sind solche Aspekte zu finden. Dieser Ansatz geht von einem intrinsisch motivierten Menschenbild aus [Donaldson & Davis 91].
Der Analyse liegt das theoretische Konstrukt des Stewardship-Ansatzes zu Grunde, aus dem mit Hilfe der von [Schweik & Semenov 03] und [Wynn 03] durchgeführten Untersuchungen zu Coworkertypen und Lebenszyklusphasen Ansätze für ein Governance-Modell entwickelt wurden. Jüngere Trends zeigen zudem, dass eine Verschärfung des Wettbewerbs einzelner Projekte um die sich verknappende Ressource „Coworker“ bereits eingesetzt hat und somit der Governance in Open-Source-Projekten eine wachsende Bedeutung zukommt [Lattemann & Stieglitz 05].
Das Ziel dieses Beitrags ist es, zentrale Open-Source-spezifische Governance-Strukturen und Koordinationsmuster sowie deren Veränderungen in verschiedenen Lebenszyklusphasen zu identifizieren, um in einem weiteren Forschungsbeitrag Module für ein Referenzmodell für die institutionelle Steuerung in diesen NPOs erstellen zu können.
Kapitel 2 beschreibt die governance-relevanten Rahmenbedingungen von Open-Source-Projekten. Hierzu gehören die Darstellung der heterogenen Zusammensetzung der Community, die Phasen des Lebenszyklus eines Open-Source-Projekts sowie die Motivationen der Mitglieder zur aktiven Teilnahme.
Kapitel 3 beschreibt Open-Source-spezifische Ansätze zur Überwachung, Kontrolle und Steuerung in Open-Source-Projekten. In diesem Kontext werden beispielhaft ausgewählte Steuerungsinstrumente dargestellt, die als Grundlage für die Entwicklung eines Maßnahmenkatalogs dienen sollen.

2. Rahmenbedingungen von Open-Source-Projekten

2.1 Charakteristika des Open-Source-Produkts und der Produkterstellung

Zur detaillierten Betrachtung von Governancestrukturen innerhalb einzelner Open-Source-Projekte werden zunächst die spezifischen Eigenschaften der freien Softwareentwicklung herausgearbeitet. Der Produktionsprozess bei der Open-Source-Software unterscheidet sich von der kommerziell erstellten Softwareproduktion vor allem durch ihren NPO-Charakter [Achtenhagen et al. 03; Schweik & Semenov 03].[1]
Das grundlegende Konzept der Erstellung frei verfügbarer Software sowie die freiwillige, in den meisten Fällen nicht entlohnte Mitarbeit durch Community-Mitglieder, wirkt direkt auf die Ausgestaltung der Organisationsstrukturen. Auch wenn

[1] Im Rahmen des Beitrags wird Open-Source-Software entsprechend der Definition der Open-Source-Initiative (OSI) verstanden, diese beinhaltet beispielsweise Regelungen zur freien Verfügbarkeit des Quellcodes. Siehe http://www.opensource.org/.

in der jüngeren Vergangenheit die Bedeutung von Coworkern, die von profitorientierten Unternehmen bezahlt und abgestellt werden [Lattemann & Stieglitz 05], gewachsen ist, soll deren Steuerung hier nicht thematisiert werden.

In Open-Source-Projekten wird auf klassische Steuerungsmechanismen, wie Arbeitszeit- oder Zielvorgaben, verzichtet. Trotz dieser Unterschiede gelingt es Open-Source-Projekten markt- und konkurrenzfähige Produkte herzustellen, wie die Beispiele Linux, Apache oder Mozilla / Firefox beweisen. Grundlage hierfür sind Spezifika, die die Entwicklung von (Open-Source-) Software aufweist. Sie unterscheidet sich von der Erstellung anderer Güter, wie Industrieprodukten und den meisten Dienstleistungen, durch folgende Merkmale, die im entscheidenden Maße positiv die Motivation der Projekt-Coworker beeinflussen, indem sie intrinsische und nicht finanzielle extrinsische Motivation verstärkt:

- Die Kommunikation und Koordination in Softwareprojekten basiert auf der Nutzung moderner Informations- und Kommunikationstechnologien (IKT)[2] und ermöglicht eine räumlich getrennte Entwicklung des Gemeinschaftswerks.
- Die zur Weiterentwicklung der Software nötigen Arbeitsbeiträge weisen einen sequenziellen Charakter auf: Innovationen bauen auf vorhergehenden Entwicklungen auf und erfolgen nicht in radikalen Sprüngen. Zudem ist die Art der Arbeitsleistung komplementär, da mit steigender Anzahl von Lösungsansätzen Synergien zur Optimierung des Produkts entstehen [Osterloh et al. 03].
- Softwareprogrammierung lässt sich in Teilprojekte zerlegen, die unabhängig voneinander weiterentwickelt werden können (Granularität) [Benkler 02].
- Softwareentwicklung kann zeitlich versetzt erfolgen. Das heißt, nachträglich programmierte Elemente oder Module können in bereits aktive Software integriert werden (Modularität) [Benkler 02].

2.2 Coworkerstrukturen in Open-Source-Projekten

Open-Source-Projekte bestehen aus heterogenen Teilgruppen, deren Mitglieder / Coworker unterschiedliche Funktionen innerhalb des Projekts wahrnehmen [Jungwirth & Franck 02, Osterloh et al. 03]. Einzelne Individuen können diesen Gruppen nur schwer eindeutig zugeordnet werden, da Open-Source-Mitglieder zumeist mehreren Gruppen gleichzeitig angehören und über die Zeit ihre Funktionen – und damit ihre Gruppenzugehörigkeit – wechseln. Auf aggregierter Ebene lassen sich drei Funktionstypen identifizieren: Bug-Fixer, Programmierer und Koordinatoren bzw. Manager [Jungwirth & Franck 02].

[2] Nachteile, die sich aus der überwiegenden Nutzung von IuK-Technologien ergeben (Anonymisierung) werden beispielsweise bei [Zimmermann 03] diskutiert.

Bug-Fixer partizipieren nur unregelmäßig und selten im Rahmen eines Open-Source-Projekts. Untersuchungen von [Raymond 99] zeigen, dass sie dennoch die größte Gruppe von Coworkertypen innerhalb der Community darstellen. Die Funktion der Bug-Fixer besteht in dem Entdecken, der Kommunikation und der Entwicklung von Lösungsvorschlägen von Programmfehlern (Beta-Testings) [u.a. Raymond 99].
Die Funktionsgruppe der **Programmierer** entwickelt die Software durch ihren Leistungsbeitrag in funktionaler Hinsicht weiter, indem sie direkte Veränderungen am Quellcode vornehmen bzw. vorschlagen. Programmierer leisten häufiger und regelmäßiger Beiträge als Bug-Fixer und sind daher intensiv in die Koordinations- und Kommunikationsstrukturen der Organisation eingebunden.
Die Gruppe der **Koordinatoren** setzt sich zumeist aus den ehemaligen Gründern und Programmierern zusammen, die sich an strategischen Überlegungen und Koordinationsaufgaben beteiligen. In großen Organisationen, wie Apache, Linux oder Perl werden Koordinatoren, durch die Gesamtheit oder einen Teil der registrierten Mitglieder gewählt. Dies verschafft den Koordinatoren eine hohe Reputation und Akzeptanz innerhalb des Projekts. Zu ihren Aufgaben zählt die Strategieentwicklung, das Change-Management und das Personalmanagement, also die institutionelle Steuerung (Governance) in Projekten.

2.3 Lebenszyklusabschnitte in Open-Source-Projekten

Durch Veränderungen, die durch den Projektlebenszyklus hervorgerufen werden, können sich divergierende, bzw. sich konterkarierende Motive der Teilgruppen herausbilden, die dazu führen, dass ein Projekt in eine Stagnation gerät. Es ist daher notwenig, die Projektphasen bzw. den Wandel im Lebenszyklus stetig zu verfolgen.
Die Identifikation von Phasen in Open-Source-Projekten ist nicht trivial. Bei klassischen Unternehmen wird der Lebenszyklus anhand von Umsatzzahlen bzw. Verkäufen identifiziert. Da Open-Source-Projekte frei verfügbare Waren produzieren, muss zur Identifikation der Lebenszyklusphasen auf andere Proxyvariablen zurückgegriffen werden. Die Veränderung der Anzahl der Downloads stellt einen geeigneten – wenn auch stark vereinfachenden – Indikator dar [Wynn 03].[3]
Es lassen sich vier, eindeutig unterscheidbare, sequenziell angeordnete Lebenszyklusabschnitte unterscheiden [Wynn 03; Schweik & Semenov 03]: (1) die

[3] Andere Indikatoren, die beispielsweise von Schweik und Semenov vorgeschlagen werden, sind: (1) jährlicher Zuwachs der Mitgliederanzahl, (2) jährlicher Zuwachs der Programmanwender, (3) Zuwachs des „Marktanteils“, (4) Anwenderzufriedenheit mit dem Produkt und (5) Funktionsumfang des Programms im Vergleich zu Konkurrenzprodukten. Im Gegensatz zur Verwendung der Downloadzahlen sind die weiteren genannten Kriterien nur mit einem erheblich größeren Aufwand messbar [Schweik & Semenov 03].

Gründungs- oder Initiierungsphase, (2) die Wachstumsphase, (3) Reifephase und (4) Niedergang oder Wiederbelebung (siehe Abbildung 1).

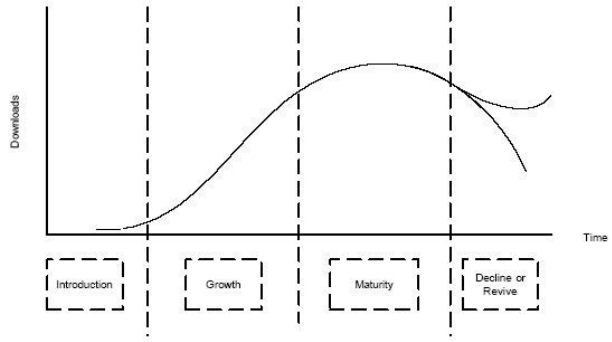

Abbildung 1: Lebenszyklusphasen [Wynn 03]

Mit jeder einzelnen Lebenszyklusphase verändern sich auch der Fokus, die Struktur, der Grad der Arbeitsteilung und die Koordinationsform innerhalb des Open-Source-Projekts (siehe Tabelle 1).

	Gründung	**Wachstum**	**Reife**	**Niedergang**
Fokus	Ideenentwicklung	Wachstum	Stabilität	Adaption
Struktur	Anarchisch, informell	Zentralisierte Steuerung	Dezentrale Steuerung, formell	Teilweise formell, aber nicht akzeptiert
Arbeitsteilung	Generalisten	Gering spezialisiert	Stark spezialisiert	Gering spezialisiert
Koordination	Informell, persönlich	Beginnende Technologieeinführung	Formell, intensive Technologienutzung	Formell, aber nicht akzeptiert

Tabelle 1: Lebenszyklusphasen in OSC [angelehnt an Wynn 03]

In der **Gründungsphase** ist es notwendig, dass die Gründer die Vision des Projektes an potenzielle Neumitglieder vermitteln, um mehr Coworker zu gewinnen. In dieser Phase wird in der Regel noch keine marktfähige Software entwickelt, daher beteiligen sich Bug-Fixer nicht an dem Projekt.

In der **Wachstumsphase** bilden sich Strukturen einer klassischen Organisation heraus. Mit dem Anstieg der Anzahl von Mitgliedern wird die Einführung von Hierarchien und Organisationsstrukturen notwendig [Wynn 03]. Die Gründer des Projekts nehmen durch die steigende Anzahl der Teilnehmer, zusätzlich zur Programmiertätigkeit, in verstärktem Maße Steuerungsaufgaben wahr [Schweik & Semenov 03].

Die **Reifephase**, in der die Zahl der Downloads nur noch langsam steigt bzw. stagniert, ist geprägt durch fest implementierte Kommunikationsstrukturen und Hierarchien [Wynn 03]. Die Koordinatoren, die sich oftmals aus den Gründern rekrutieren, nehmen

in erster Linie strategische Aufgaben wahr. Diese umfassen Entscheidungen über die mittel- bis langfristige Entwicklung des Programms.

Der **Niedergang** von Open-Source-Projekten kann bereits nach jeder der vorangegangenen Phasen einsetzen und unterschiedliche Ursachen haben. Beispiele hierfür sind, dass das Produkt durch andere Neuentwicklungen obsolet geworden ist, dass die ehemaligen Gründer das Projekt verlassen und keine Nachfolger gefunden werden können oder dass die Führer des Projekts die Software in eine Richtung entwickeln möchten, die durch die Mehrheit der Programmierer und Bug-Fixer nicht getragen wird. Es kommt zu Interessenskonflikten [Wynn 03].

Gelingt es nicht, diesen Faktoren entgegenzusteuern, verliert das Projekt, aufgrund der schwindenden Motivation der Mitglieder, weiter an Bedeutung. Wie [Wynn 03] sowie [Schweik und Semenov 03] feststellen, besteht jedoch auch die Möglichkeit einer Wiederbelebung, wenn die vorhandenen Probleme beseitigt werden. Dies kann vor allem durch den gezielten Einsatz von Steuerungsinstrumenten gefördert werden.

2.4 Motivation und Ziele von Open-Source-Coworkern

2.4.1 Rolle der Motivation in Open-Source-Projekten

Grundsätzlich werden zwei Arten der Motivation, die die Coworker zur Mitarbeit bewegen, unterschieden: Zum einen die extrinsische Motivation, die in klassischen Unternehmen eine dominante Stellung einnimmt [Shah 03]. Hierunter wird ein Arbeits- oder Lernanreiz, der durch die Erwartung nachfolgender Belohnung (z.B. Verbesserung der Karrierechancen) ausgelöst wird, verstanden. Ein solcher Anreiz kann materiell, beispielsweise in Form einer finanzieller Entlohnung, oder durch soziale Anerkennung der Personen signalisiert werden. Die intrinsiche Motivation zum anderen basiert im Gegensatz hierzu nicht auf äußeren Anreizen, sondern entsteht aus dem Individuum selbst.

Diverse wissenschaftliche Beiträge haben sich mit der Motivation in Open-Source-Projekten beschäftigt. Dabei sind zentrale intrinsische Motive erkannt worden, wie etwa der Spaß und das Verlangen etwas zu entwickeln oder zu verbessern [Shah 03]. Daneben sind Motive extrinsischer Natur identifiziert worden, wie etwa der Wunsch zur eigenen Nutzung der Software [Osterloh et al. 03], die Erhöhung der Reputation in einer Gruppe, Zugehörigkeitsbedürfnis zu einer Peer Group, Identitäts- und Imageaufbau, Vermittlung von Werten und Ideologien sowie die Nachfrage nach Trainings, Karriere und Signaling der eigenen Fähigkeiten an externe Stellen [Raymond 99; Lerner & Tirole 01]. Im Rahmen der Analyse wird diesen Autoren gefolgt und davon ausgegangen, dass auf Open-Source-Coworker sowohl intrinsische als auch extrinsische Anreize wirken, die sich jedoch im Laufe des Lebenszyklus verändern können.

2.4.2 Wandel der Motivation im Lebenszyklus des Open-Source Projekts

Bezieht man die Heterogenität der Mitglieder sowie den lebenszyklusbedingten Wandel von Open-Source-Projekten ein, so sind weitere Schlüsse über die Motivation der Mitglieder ableitbar. Mit dem Wandel in der grundlegenden Zielsetzung einiger Projekt-Mitglieder über die Phasen des Projektlebenszyklus geht eine Veränderung der Motivation zur Zuarbeit durch die Mitglieder einher. Die Verbesserung der Software bleibt mit fortgeschrittenem Projektverlauf somit nicht mehr das eigentliche Ziel, sondern für die Koordinatoren und Programmierer ein Mittel, um der Organisation mehr Mitglieder zuzuführen und die externe Reputation sowie das eigene Fachwissen zu erhöhen. Diese Ziele dienen wiederum der Verbesserung der eigenen Karrierechancen. Der Aspekt des Signaling wurde insbesondere von [Lerner & Tirole 01] herausgestellt und untersucht.

Die Teilnahme der **Bug-Fixer** an Open-Source-Projekten wird ursprünglich durch das Interesse an der Verbesserung der Software motiviert [Lerner & Tirole 01]. Die Möglichkeit, auf höherer Ebene an dem Projekt mitzuarbeiten, stellt ein weiteres Motiv dar. Der Anreiz einer dadurch induzierten Steigerung der Reputation wird durch die Bekanntgabe der produktivsten Bug-Fixer ebenfalls angesprochen [Lerner & Tirole 01].

Zahlreiche Untersuchungen deuten darauf hin, dass Open-Source-Teilnehmer das Programmieren als individuellen kreativen Beitrag zu einer übergeordneten nützlichen Sache sehen [Lakhani & Hippel 01]. Die Programmierarbeit selbst erzeugt eine Art Selbstverwirklichungsgefühl und Spaß und wird daher von vielen nicht als Arbeit sondern eher als Freizeitbeschäftigung eingestuft [Achtenhagen et al. 03, Osterloh et al. 03]. Die Motivation der **Programmierer** ist in der Gründungsphase überwiegend intrinsischer Natur. Schreitet das Projekt weiter fort, verfolgen Entwickler und Programmierer zusätzlich das Ziel einer höheren internen (innerhalb der Open Source Community) und externen (über die Community hinaus) Reputation, und damit einhergehend, besseren Karrierechancen [Osterloh et al. 03].

Die Arbeitsmotivation der **Koordinatoren** ist im späteren Verlauf des Projekts unter anderem auf Karriereüberlegungen zurückzuführen [Lerner & Tirole 01]. In der Initiierungsphase sind die Gründer jedoch vorwiegend intrinsisch motiviert. Die intrinsische Motivation kann im Laufe des Lebenszyklus abnehmen, da sich das Aufgabenfeld der Koordinatoren mit dem Projektlebenszyklus deutlich wandelt.

3. Governance in Open-Source-Projekten

3.1 Notwendigkeit der Governance in Open-Source-Projekten

Wie gezeigt wurde, handelt es sich bei einer Open-Source-Gemeinschaft um ein komplexes und prekäres System auf der Basis auszubalancierender Interessenslagen. Hieraus ergibt sich, dass die Organisation nur dann dauerhaft Bestand hat, wenn sich für jeden Teilnehmer zumindest ein Gleichgewicht aus Anreizen und Aufwendungen ergibt. Verändern sich jedoch im Laufe der Zeit die Motivationsgrundlagen der Mitglieder, so müssen daher auch entsprechend neue Anreize geboten bzw. angepasst werden, um das Gleichgewicht zu erhalten. Besteht ein Bedarf zur Anpassung von Steuerungsmechanismen, obliegt es den Koordinatoren mögliche Alternativen zu entwickeln und umzusetzen.

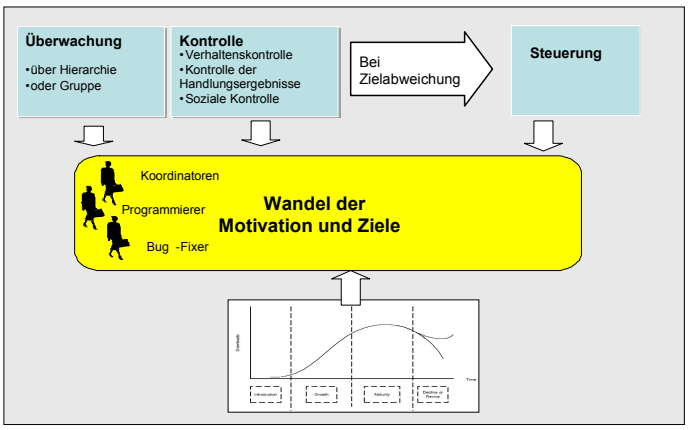

Abbildung 2: Governance in Open-Source-Projekten

Abbildung 2 verdeutlicht, dass sich mit dem Lebenszyklus Ziele und Motive der drei Coworkertypen ändern. Mithilfe von Überwachung und Kontrolle, die beide Zieldefinitionen voraussetzen, kann mittels einer adäquaten Steuerung – im Falle der Abweichung vom Soll – den Zielen und Werten der Coworker entsprochen werden.

3.2 Überwachung in Open-Source-Projekten

Die Überwachung dient der frühzeitigen Erkennung von Risiken. In Open-Source-Projekten findet die Überwachung regelmäßig nicht nur durch eine hierarchisch übergeordnete Führungsstruktur statt, sondern auch durch das Kollektiv der partizipierenden Coworker selbst. Dies geschieht beispielsweise durch laufendes Begutachten und Bewerten neuer Programmbeiträge durch die in dem Projekt beschäftigten Programmierer (Peer-Reviewing) [Benkler 02].

Weiterhin nimmt die Gruppe der Koordinatoren direkte Überwachungsfunktionen innerhalb des Projektes wahr. Ihre Aufgabe ist es, die Bedürfnisse der einzelnen

Mitglieder bzw. Mitgliedergruppen zu erfassen und nach Möglichkeit zu befriedigen, um den Motivationserhalt sicherzustellen. Es ist daher von hoher Bedeutung, festzustellen, durch welche Anreize neue Mitglieder gewonnen bzw. die schon vorhanden Mitglieder zu einer Erhöhung ihrer Arbeitsbeiträge veranlasst werden können. Da sich die Bedürfnisse der Mitgliedergruppen in den einzelnen Lebenszyklusabschnitten verändern, können die Koordinatoren durch die Identifizierung der aktuellen Lebensabschnittsphase Rückschlüsse auf die zu bietenden Anreize ziehen, was jedoch lediglich als grobe Orientierung dienen kann, da die Projektlebensphasen einem kontinuierlichen Wandel unterzogen sind.

3.3 Kontrolle in Open-Source-Projekten

Traditionelle Kontrollformen (behavorial und output control) werden in der betriebswirtschaftlichen Organisationslehre ausführlich und zumeist einheitlich behandelt. Diese Kontrollformen lassen sich auch in Open-Source-Gemeinschaften partiell umsetzen. Dem Konzept der Sozialen Kontrolle durch die Bindung der Community kommt eine erhebliche Bedeutung zu. Grundsätzlich stehen drei Kontrollmöglichkeiten in Open-Source-Projekten zur Verfügung [Lattemann & Köhler 04], die von der Community selbst oder von den Koordinatoren ausgeübt werden können:

- Leistungsermittlung und Beurteilung in Form der direkten Verhaltenskontrolle (behavioral control) anhand von aus Erfahrungen gewonnenen Standards.
- Indirekte, am Handlungsergebnis orientierte Kontrolle (output control) mithilfe von Zielen und eines Soll-Ist-Vergleichs.
- Soziale Kontrolle (social control), die überprüft, inwieweit Mitglieder gleiche Wert- und Zielvorstellungen haben.

Die Soziale Kontrolle basiert vor allem auf dem Konzept des Vertrauens [Lattemann & Köhler 04].[4] Im Falle der Open-Source-Gemeinschaften müssen die Teilnehmer darauf vertrauen, dass die von ihnen geleisteten Arbeitsbeiträge im Sinne des Projekts verwendet werden. Die Genese von Vertrauen in und zwischen Organisationen kann durch soziale Normen und institutionelle Rahmenbedingungen gezielt gefördert werden.

3.4 Steuerung in Open-Source-Projekten

Ergibt sich aus der Analyse der Überwachungs- und Kontrolltätigkeiten ein Bedarf für Veränderungen, so können diese mittels verschiedener Steuerungsmaßnahmen umgesetzt werden. In den einzelnen Lebensphasen sind verschiedene Governance-

[4] Dabei ist Vertrauen die „freiwillige Erbringung einer riskanten Vorleistung unter Verzicht auf explizite vertragliche Sicherungs- und Kontrollmaßnahmen gegen opportunistisches Verhalten ...“ [Ripperger 98].

Instrumente unterschiedlich effektiv, um den Zusammenhalt der Organisation und eine erfolgreiche Weiterentwicklung des Softwareprodukts zu gewährleisten.
Maßnahmen sind insbesondere zu ergreifen, wenn mit der steigenden Projektkomplexität und steigender Teilnehmerzahl die Informationsasymmetrien zunehmen und zu erhöhten Informations- und Transaktionskosten führen [Achtenhagen et al. 03].
Ein Verzicht auf die Implementierung von Governance-Instrumenten kann das Wachstum und die Effizienz der Organisation behindern bzw. das Projekt frühzeitig durch divergierende Interessenslagen in eine Stagnation oder in den Niedergang führen.
Weiterhin wirkt die falsche Auswahl von Steuerungsinstrumenten dann kontraproduktiv, wenn durch die Aktivierung extrinsischer Motive die intrinsische Motivation in einem solchen Maße abnimmt, dass insgesamt ein negativer Effekt entsteht [Osterloh et al. 03; Dilger 2004].
Folgende Governance-Instrumente, deren Einsatz sich in einzelnen Open-Source-Projekten beobachten lässt, können unter Einbezug der Rahmenbedingungen dazu beitragen, eine effektive Steuerung zu betreiben:

- Einführung oder Anpassung von Hierarchien
- Einführung oder Anpassung von Informations- und Kommunikationstechnologien
- Schaffung zusätzlicher Anreize (wie etwa regelmäßige und persönliche Treffen, Schulungen, Einführung neuer Teilprojekte)
- Veröffentlichung von Reputationslisten und Sanktionierung von Opponenten mittels „flaming“
- Schaffung von Privilegien (bspw. Teilnahme an bestimmten Projekten, Zugriff auf Code Repository)
- Einführung eines Ethikkodexes, der der Kommunikation von Normen dient[5]

Da in Open-Source-Projekten nur bedingt Kennziffern zur Erfolgsmessung herangezogen werden können, ist es jedoch überaus schwierig die positive Wirkung der dargestellten Steuerungsmaßnahmen auf das Gesamtprojekt nachzuweisen.

4. Fazit

Die Darstellung zeigt, dass Open-Source-Projekte organisationstheoretischen Problemen unterliegen. Vertrauen der Mitglieder untereinander und zum Gesamtprojekt stellt eine wesentliche Erfolgsgrundlage der Wertschöpfung in Open-Source-Projekten dar. Insbesondere haben die unterschiedlich ausgeprägten Motive der Open-Source-Mitglieder und deren Wandel über die Zeit einen erheblichen Einfluss auf den Erfolg von Open-Source-Projekten. Der Einsatz von Governance-Instrumenten muss noch

[5] Siehe bspw. http://www.apache.org/foundation/bylaws.html

stärker als in gewinnorientierten Unternehmen die individuellen intrinsischen und extrinsischen Motivationsfaktoren der Teilnehmer berücksichtigen. Gleichzeitig muss sich die Auswahl der Governance-Instrumente an veränderte Bedingungen und Anforderungen anpassen.

Neben den dargestellten zentralen Motivatoren in Open-Source-Projekten sind weitere Anreizmechanismen identifizierbar, die sowohl auf die intrinsische als auch auf die extrinsische Motivation abzielen. Die unterschiedlichen Beweggründe der heterogenen Community-Mitglieder machen eine Untersuchung von Open-Source-Projekten auf einer Mikroebene notwendig.

Auf Grundlage der dargestellten Erkenntnisse wird in weiteren Arbeiten ein Rahmenmodell für die Governance in Open-Source-Projekten erarbeitet und der Einsatz elektronischer Medien im Governance-Prozess untersucht.

Literatur

[Achtenhagen et al. 03] Achtenhagen, L., Müller-Lietzkow, J., Knyphausen-Aufseß, D. zu (2003): Das Open-Source-Dilemma: Open-Source-Software zwischen freier Verfügbarkeit und Kommerzialisierung, Schmalenbachs Zeitung für betriebswirtschaftliche Forschung, Düsseldorf, S. 455-481.

[Benkler 02] Benkler, Y. (2002): Coase`s Penguin, or, Linux and the Nature of the Firm, The Yale Law 112 (3).

[Dilger 04] Dilger, F. (2004): Negative Wirkungen verstärkter Anreize aus ökonomischer Sicht. Journal für Betriebswirtschaft 1/2004, Linder Verlag Wien.

[Donaldson & Davis 91] Donalsdon, L, Davis, J.H. (1991): Stewardship theory or agency theory: CEO governance and shareholder returns, Australian Journal of Management 16.

[Jungwirth &Franck 02] Jungwirth, C., Franck, E. (2002): Open versus Closed Software – Eine organisationsökonomische Betrachtung zum Wettbewerb der Betriebssysteme Windows und Linux, Universität Zürich.

[Lakhani & Hippel 01] Lakhani, K.R., E. von Hippel (2001): How Open-Source-Software Works: “free” user-to-user assistance, Research Policy 32, Cambridge.

[Lattemann & Köhler 04] Lattemann, C., Köhler, T. (2004): Vertrauen ist gut – Kontrolle ist besser?. In: Eicker et al. (Hrsg.): Multikonferenz Wirtschaftsinformatik Band I, Infix, S. 306-323.

[Lattemann & Stieglitz 05] Lattemann, C., Stieglitz, S. (2005): Co-Worker Governance in Open-Source Projects, erscheint in: Bitzer und Schröder (Hrsg.), The Economics of Open-Source-Software Development Analyzing Motivation, Organization, Innovation and Competetion in the Open-Source-Software Revolution. Elsevier.

[Lerner & Tirole 01] Lerner, J., Tirole, J. (2001): Some Simple Economics of Open-Source, The Journal of Industrial Economics 50, S. 197-234.

[Osterloh et al 03] Osterloh, M., Rota, S., Kuster, B. (2003): Open-Source-Software Produktion: Ein neues Innovationsmodell?, Forschungsarbeit am Institut für betriebswirtschaftliche Forschung der Universität Zürich.

[Raymond 99] Raymond, E. S. (1999), The Magic Cauldron: http://www.catb.org/~esr/writings/magic-cauldron/., Abruf am 2004-07-10.

[Ripperger 98] Ripperger, T. (1998): Ökonomik des Vertrauens. Analyse eines Organisationsprinzips, Tübingen: Mohr.

[Schneider & Kenis 96] Schneider, V., Kenis, J. (1996): Verteilte Kontrolle – Institutionelle Steuerung in modernen Gesellschaften, in: Kenis, J., Schneider, V. (Hrsg.): Organisation und Netzwerke. Institutionelle Steuerung in Wirtschaft und Politik.

[Schweik &Semenov 03] Schweik, C.M., Semenov, A. (2003): The Institutional Design of Open-Source Programming: Implications for Adressing Complex Public Policy and Management Problems, First Monday 8 (1), Chicago.

[Shah 03] Shah, S. (2003): Understanding the Nature of Participation & Coordination in Open and Gated Source Software Development Communities, MIT Sloan School of Management, Working Paper.

[Wynn 03] Wynn, D.E. (2003): Organizational Structure of Open-Source Projects: A Life Cycle Approach, Abstract for 7th Annual Conference of the Southern Association for Information Systems, Georgia.

[Zimmermann 03] Zimmermann, F. (2003): Vertrauen in Virtuellen Unternehmen. In der Reihe: Szyperski et al. (Hrsg.): Planung, Organisation und Unternehmensführung, Band 85, Josef Eul Verlag, Köln.